#2

520.3963200

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant(s): FURUYA, ET AL.

Serial No.:

Filed: February 16, 2001

Title: METHOD AND APPARATUS FOR SYMMETRIC-KEY ENCRYPTION

Group:

LETTER CLAIMING RIGHT OF PRIORITY

Honorable Commissioner of
Patents and Trademarks
Washington, D.C. 20231

520.39632X00

Sir:

Under the provisions of 35 USC 119 and 37 CFR 1.55, the applicant(s) hereby claim(s) the right of priority based on Japanese Patent Application No.(s) 2000-070994 filed March 9, 2000 and 2000-210690 filed July 6, 2000.

Certified copies of said Japanese Application are attached.

Respectfully submitted,

ANTONELLI, TERRY, STOUT & KRAUS, LLP

Carl I. Brundidge
Registration No. 29,621

CIB/mdt
Attachment
(703)312-6600

日 本 国 特 許 庁
PATENT OFFICE
JAPANESE GOVERNMENT

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。
This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日
Date of Application: ２０００年 ３月 ９日

出 願 番 号
Application Number: 特願２０００－０７０９９４

出 願 人
Applicant (s): 株式会社日立製作所

CERTIFIED COPY OF
PRIORITY DOCUMENT

２００１年 １月１２日

特 許 庁 長 官
Commissioner,
Patent Office 及川耕造

出証番号 出証特２０００－３１１１６５８

【書類名】　特許願

【整理番号】　K00000351

【提出日】　平成12年 3月 9日

【あて先】　特許庁長官殿

【国際特許分類】　G09C 1/00

【請求項の数】　30

【発明者】

【住所又は居所】　神奈川県川崎市麻生区王禅寺１０９９番地　株式会社日立製作所　システム開発研究所内

【氏名】　古屋　聡一

【発明者】

【住所又は居所】　神奈川県川崎市麻生区王禅寺１０９９番地　株式会社日立製作所　システム開発研究所内

【氏名】　宝木　和夫

【発明者】

【住所又は居所】　神奈川県横浜市戸塚区戸塚町５０３０番地　株式会社日立製作所　ソフトウェア事業部内

【氏名】　車谷　博之

【発明者】

【住所又は居所】　神奈川県川崎市麻生区王禅寺１０９９番地　株式会社日立製作所　システム開発研究所内

【氏名】　髙橋　昌史

【発明者】

【住所又は居所】　神奈川県川崎市麻生区王禅寺１０９９番地　株式会社日立製作所　システム開発研究所内

【氏名】　宮崎　邦彦

【発明者】

【住所又は居所】　神奈川県川崎市麻生区王禅寺１０９９番地　株式会社日立製作所　システム開発研究所内

【氏名】　佐藤　尚宜

【特許出願人】

【識別番号】　000005108

【氏名又は名称】　株式会社日立製作所

【代理人】

【識別番号】　100075096

【弁理士】

【氏名又は名称】　作田　康夫

【手数料の表示】

【予納台帳番号】　013088

【納付金額】　21,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　共通鍵暗号方法及び装置

【特許請求の範囲】

【請求項１】

共通鍵暗号方法であって，

冗長データとメッセージとからなる平文を特定の長さで区切った複数の平文ブロックを生成し，

秘密鍵から，乱数列を生成し，前記乱数列から前記平文ブロックに対応する乱数ブロックを生成し，

前記平文ブロックと前記乱数ブロックとを演算して得た他の前記平文ブロックへのフィードバック値を出力し，

前記平文ブロックと，前記乱数ブロックと，他の平文ブロックの演算から得られるフィードバック値とを用いて暗号文ブロックを暗号演算する共通鍵暗号方法。

【請求項２】

請求項1記載の共通鍵暗号方法であって，

前記暗号演算は一つ以上の，その合計の長さは，前記暗号文ブロックより長い前記乱数ブロックを用いる共通鍵暗号方法。

【請求項３】

請求項２記載の共通鍵暗号方法であって，

前記暗号演算として，前記平文ブロックを用いた2項演算や単項演算を決められた手順に従って1回以上行い，

得られた複数の暗号文ブロックを組み合わせ，暗号文として出力する共通鍵暗号方法。

【請求項４】

請求項２記載の共通鍵暗号方法であって，

前記暗号演算を有限体上の乗算と加算により行う共通鍵暗号方法。

【請求項５】

請求項２記載の共通鍵暗号方法であって，

前記暗号演算を巡回シフト演算と,算術乗算との組み合わせによって行う共通鍵暗号方法。

【請求項６】

請求項２記載の共通鍵暗号方法であって，

擬似乱数生成手段を備え，前記秘密鍵から前記乱数列を生成する共通鍵暗号方法。

【請求項７】

請求項６の共通鍵暗号方法であって，

前記メッセージを複数に分割し，

前記擬似乱数生成器を用いて，分割数分の乱数列を生成し，

前記分割したメッセージと前記乱数列のいずれかをそれぞれ異なる演算ユニットに割り当て，並列処理を行う共通鍵暗号方法。

【請求項８】

共通鍵暗号の復号方法であって，

暗号文を特定の長さで区切った複数の暗号文ブロックを生成し，

秘密鍵から，乱数列を生成し，前記乱数列から前記暗号文ブロックに対応する乱数ブロックを生成し，

前記暗号文ブロックと前記乱数ブロックとを演算して得た他の前記暗号文ブロックへのフィードバック値を出力し，

前記暗号文ブロックと，前記乱数ブロックと，他の暗号文ブロックの演算から得られるフィードバック値と，を用いて平文ブロックを復号演算する共通鍵暗号の復号方法。

【請求項９】

請求項８記載の共通鍵暗号の復号方法であって，

前記復号演算は一つ以上の，その合計の長さは，前記暗号文ブロックより長い前記乱数ブロックを用いる共通鍵暗号の復号方法。

【請求項１０】

請求項９記載の共通鍵暗号の復号方法であって，

前記平文ブロックを複数連結して平文を生成し，

前記平文に含まれる冗長データを抽出し，

前記冗長データを検査し，前記暗号文への改ざんの有無を検出する共通鍵暗号の復号化方法。

【請求項１１】

共通鍵暗号装置であって，

冗長データとメッセージとからなる平文を入力し，特定の長さで区切った複数の平文ブロックを生成する回路と，

秘密鍵を入力し，乱数列を生成し，前記乱数列から前記平文ブロックに対応する乱数ブロックを生成する回路と，

前記平文ブロックと前記乱数ブロックとを演算して得た他の前記平文ブロックへのフィードバック値を出力する回路と，

前記平文ブロックと，前記乱数ブロックと，他の平文ブロックと他の乱数ブロックとの演算によって得られるフィードバック値と，を用いて暗号文ブロックを暗号演算する回路と

を備える共通鍵暗号装置。

【請求項１２】

請求項１１記載の共通鍵暗号装置であって，

前記暗号演算回路は，一つ以上の，その合計の長さは，前記暗号文ブロックより長い前記乱数ブロックを用いる，共通鍵暗号装置。

【請求項１３】

請求項１２記載の共通鍵暗号装置であって，

前記暗号演算回路は，前記平文ブロックを用いた2項演算または単項演算を決められた手順に従って1回以上行う回路を含み，

得られた複数の暗号文ブロックを組み合わせ，暗号文として出力する回路を備える共通鍵暗号装置。

【請求項１４】

請求項１２記載の共通鍵暗号装置であって，

前記暗号演算回路は，有限体上の乗算と加算を行う共通鍵暗号装置。

【請求項１５】

請求項１２記載の共通鍵暗号装置であって，

前記暗号演算回路は，巡回シフト演算回路と算術乗算回路とを備える共通鍵暗号装置。

【請求項１６】

請求項１２記載の共通鍵暗号装置であって，

擬似乱数生成器を備え，前記秘密鍵から前記乱数列を生成する共通鍵暗号装置。

【請求項１７】

請求項１６の共通鍵暗号装置であって，

前記メッセージを複数に分割する回路と，

前記擬似乱数生成器を用いて，分割数分の乱数列を生成する回路と，

複数の演算ユニットと，

前記分割したメッセージと前記乱数列のいずれかをそれぞれ異なる演算ユニットに割り当てる回路と，

を備える共通鍵暗号装置。

【請求項１８】

共通鍵暗号の復号装置であって，

暗号文を入力し，特定の長さで区切った複数の暗号文ブロックを生成する回路と，

秘密鍵を入力し，前記暗号文よりも長い乱数列を生成し，前記乱数列から前記暗号文ブロックに対応する乱数ブロックを生成する回路と，

前記暗号文ブロックと前記乱数ブロックとを演算して得た他の前記暗号文ブロックへのフィードバック値を出力する回路と，

前記暗号文ブロックと，前記乱数ブロックと，他の暗号文ブロックの演算から得られるフィードバック値と，を用いて平文ブロックを復号演算する回路と

を備える共通鍵暗号の復号装置。

【請求項１９】

請求項１８記載の共通鍵暗号の復号装置であって，

前記復号演算回路は，一つ以上の，その合計の長さは，長い前記乱数ブロックを用いる，共通鍵暗号の復号装置。

【請求項２０】

請求項１９記載の共通鍵暗号の復号装置であって，

前記平文ブロックを複数連結して平文を生成する回路と，

前記平文に含まれる冗長データを抽出する回路と，

前記冗長データを検査して前記暗号文への改ざんの有無を検出する回路と

を備える共通鍵暗号の復号装置。

【請求項２１】

コンピュータに読み込まれ，共通鍵暗号方法を実行させるプログラムを記憶した媒体であって,前記プログラムは，前記コンピュータに，

冗長データとメッセージとからなる平文を読み込ませ，前記平文を特定の長さで区切った複数の平文ブロックを生成させ，

秘密鍵を入力させ，乱数列を生成させ，前記乱数列から前記平文ブロックに対応する乱数ブロックを生成させ，

前記平文ブロックと前記乱数ブロックとを演算させて得た他の前記平文ブロックへのフィードバック値を出力させ，

前記平文ブロックと，前記乱数ブロックと，他の平文ブロックと他の乱数ブロックとの演算によって得られるフィードバック値と，を用いて暗号文ブロックを暗号演算させる

プログラムを記憶した媒体。

【請求項２２】

請求項２１記載のプログラムを記憶した媒体であって，

前記暗号演算として，一つ以上の，その合計の長さは，前記暗号文ブロックより長い前記乱数ブロックを用いらせる，

プログラムを記憶した媒体。

【請求項２３】

請求項２２記載のプログラムを記憶した媒体であって，

前記暗号演算として，前記平文ブロックを用いた2項演算または単項演算を決

められた手順に従って1回以上行なわせ，

得られた複数の暗号文ブロックを組み合わせ，暗号文として出力させるプログラムを記憶した媒体。

【請求項２４】

請求項２２記載のプログラムを記憶した媒体であって，

前記暗号演算として，有限体上の乗算と加算を行わせるプログラムを記憶した媒体。

【請求項２５】

請求項２２記載のプログラムを記憶した媒体であって，

前記暗号演算は，巡回シフト演算と算術乗算とを行わせるプログラムを記憶した媒体。

【請求項２６】

請求項２２記載のプログラムを記憶した媒体であって，

擬似乱数生成を行わせ，前記秘密鍵から前記乱数列を生成させるプログラムを記憶した媒体。

【請求項２７】

請求項２６のプログラムを記憶した媒体であって，

前記メッセージを複数に分割させ，

前記擬似乱数生成を，分割数分行わせ，

前記分割したメッセージと前記乱数列のいずれかをそれぞれ異なる演算ユニットに割り当てさせる，プログラムを記憶した媒体。

【請求項２８】

プログラムを記憶した媒体であって，

暗号文を入力し，特定の長さで区切った複数の暗号文ブロックを生成する回路と，

秘密鍵を入力し，前記暗号文よりも長い乱数列を生成し，前記乱数列から前記暗号文ブロックに対応する乱数ブロックを生成する回路と，

前記暗号文ブロックと前記乱数ブロックとを演算させて得た他の前記暗号文ブロックへのフィードバック値を出力させ，

前記暗号文ブロックと，前記乱数ブロックと，他の暗号文ブロックの演算から得られるフィードバック値と，を用いて平文ブロックを復号演算させるプログラムを記憶した媒体。

【請求項２９】

請求項２８記載のプログラムを記憶した媒体であって，

前記復号演算として，一つ以上の，その合計の長さは，長い前記乱数ブロックを用いらせる，プログラムを記憶した媒体。

【請求項３０】

請求項２９記載のプログラムを記憶した媒体であって，

前記平文ブロックを複数連結して平文を生成させ，

前記平文に含まれる冗長データを抽出させ，

前記冗長データを検査させて前記暗号文への改ざんの有無を検出させるプログラムを記憶した媒体。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は，秘密情報のセキュリティを確保する技術に関する。

【０００２】

【従来の技術】

従来の暗号処理装置は，データを秘匿する目的のブロック暗号やストリーム暗号が提案されていた。ブロック暗号にはDES，IDEAなど非常にいろいろな方式が提案されている。DES，IDEAについては以下の文献で扱っている。

文献1:Menezes, van Oorschot, Vanstone, Handbook of Applied Cryptography, CRC Press, 1996, pp.250-259, pp.263-266。

【０００３】

ブロック暗号はECB，CBC，CFB，OFB，カウンタモードなどのブロック暗号操作モードにより全体の暗号処理の安全性や性質を議論するが，これまでに暗号化処理と改ざん検出を同時に行う操作モードは，iaPCBCモードが知られているだけで，残りのモードは改ざんの検出がそれ自身では不可能である。ブロック暗号の操作モー

ドについては以下の文献で扱っている。

文献2:Schneier, Applied Cryptography, Second Edition, John Wiley & Sons, Inc。, 1996, pp.189-209。

iaPCBCモードは以下の文献で扱っている。

文献3:Gligor, Donescu, "Integrity-Aware PCBC Encryption Schemes," Preproceedings in Secure Protocol Workshop, Cambridge, 1999, to appear in Lecture Notes in Computer Science series, Springer-Verlag。

【０００４】

iaPCBCモードはブロック暗号を用いた操作モードであり,暗号化処理では並列処理が不可能,事前計算などが不可能,などの点で非常に高速な処理が要求される環境への実装が非常に困難であった。

これに対して,メッセージ認証子と呼ばれる改ざん検出のための暗号学的チェックサム(以下, MACという)を生成する方式が提案され,ブロック暗号の上記操作モードの暗号処理でも必要に応じてMAC生成処理を同時にかつまったく独立の機構として実装することで,暗号処理と改ざん検出が同時に可能となった。しかし,この場合にはまったく独立な暗号学的鍵を2度,すなわち,暗号化用と改ざん検出用,共有する必要があるという点,それから暗号化されるデータを2度処理,つまり,暗号化処理とMAC生成処理,にかける必要があり,システムが複雑になったり,長いデータの処理に向かないなどの懸念があった。さらにブロック暗号の処理速度が現在の通信の速度に比べて低速であり,これらブロック暗号とMACの組み合わせ技術は,ギガビットやテラビット処理への応用が困難であった。MACについての記述は以下の文献にある。

文献4:Menezes, van Oorschot, Vanstone, Handbook of Applied Cryptography, CRC Press, 1996, pp.352-368。

【０００５】

これに対してストリーム暗号は,さまざま提案された暗号学的擬似乱数生成器を用いて暗号化を行うメカニズムで,擬似乱数生成器の安全性や性質とは無関係に改ざんの検出はそれ自身では不可能だった。ストリーム暗号,またはストリーム暗号に用いる擬似乱数生成器としてよく知られているものに,SEAL,非線形コンバ

イナを用いた線形フィードバックシフトレジスタ,非線形フィルタを用いた線形フィードバックシフトレジスタ,クロック制御方式の線形フィードバックシフトレジスタなどがある。SEALについては以下の文献で扱っている。

文献5:Schneier, Applied Cryptography, Second Edition, John Wiley & Sons, Inc., 1996, pp.398-400。

【０００６】

その他のフィードバックシフトレジスタに基づく方式については以下の文献で扱っている。

文献6:Menezes, van Oorschot, Vanstone, Handbook of Applied Cryptography, CRC Press, 1996, pp.203-212。

【０００７】

ストリーム暗号とMACを組み合わせた技術により暗号処理と改ざん検出を同時に行えるほか,上記ブロック暗号の処理にくらべて,2倍～20倍の比率でストリーム暗号がより高速であるが,どのMAC生成方式についても,ブロック暗号とMACの組み合わせ同様,ふたつの異なる鍵の共有が必要であったり,ひとつのメッセージについて2度の処理を行う必要などがあった。より詳しくMAC生成方式を考えると,本来のストリーム暗号に,付帯的に必要となるメカニズムや計算量が非常に大きい。例えば,HMAC,UMACなどのMAC生成方式では暗号学的に衝突なしで一方向性を保証している安全なハッシュ関数を必要であり,ストリーム暗号にさらに前述のハッシュ関数を実装する必要がある。HMACは前述の文献4,pp.355のExample 9.67で扱っている。またUMACは以下の文献で扱っている。

文献7:Black, Halevi, Krawczyk, Krovetz, Rogaway, "UMAC: Fast and Secure Message Authentication," Advances in Cryptology, - CRYPTO'99, Lecture Notes in Computer Science, Vol. 1666, Springer-Verlag, 1999。

【０００８】

しかし,一般にSHA-1, MD5などのハッシュ関数は非常に複雑であり,実装も容易でない。これらのハッシュ関数については以下の文献で扱っている。

文献8:Menezes, van Oorschot, Vanstone, Handbook of Applied Cryptography, CRC Press, 1996, pp.347-349。

【０００９】

ハッシュ関数の安全性の面についても,ブロック暗号のそれに比べて議論が行われておらず,ユーザによってはハッシュ関数を信頼できず,組み込めないこともある。またMAC生成方式の中で擬似乱数生成器のみを用いる方式であるMMHのMAC生成では非常に少ない追加の資源,回路やプログラム,で暗号処理に改ざん検出の機能を付け加えることができるが,メッセージの長さに対してその2倍の長さの擬似乱数を必要とし,必要な乱数生成に時間がかかってしまう。MMHは以下の文献で扱っている。

文献9:Halevi, Krawczyk, "MMH: Software Message Authentication in the Gbit/Second Rates," Fast Software Encryption, 4th International Workshop, FSE'97, Lecture Notes in Computer Science, Vol。1267, Springer-Verlag, 1997。

【００１０】

【発明が解決しようとする課題】

従来の暗号技術のほとんどは,復号化の際,改ざん検出をそれ自身ではすることができなかった。改ざん検出を行うとき,異なるふたつの鍵共有の必要性や,独立の処理,別の暗号学的要素関数の追加実装などが必要であった。

本発明は改ざん検出も復号化と同時に可能であり,またデータ秘匿,データ改ざんに対する安全性については,証明可能なものを提供することを目的とする。

処理速度の面についての課題として,ブロック暗号のこれまで知られた操作モードでは並列度や事前計算等の可能性がなく,高並列処理や高速処理には向いていないという点があった。

本発明は擬似乱数生成器の高速処理性能を生かしながら,事前計算や並列処理の利点をもつ,共通鍵暗号の方法を提供することを目的とする。

本発明は従来のブロック暗号よりも高速な処理が可能というだけではなく,資源が許される場合にはそれだけより高速な処理が可能であり,高い並列度を達成できるような高速処理に向けたものを提供することを目的とする。

本発明は非常に短いメッセージについても処理速度が落ちないような方式を提供することを目的とする。

本発明はストリーム暗号に非常に小さな追加の回路，またはプログラムで実装可能な方式を提供することを目的とする。

【００１１】

【課題を解決するための手段】

本発明では，擬似乱数列を鍵ストリームとして用いてブロック毎に処理を行い，かつ，改ざん検出を可能にする共通鍵暗号方法を提供する。

本発明の第一の態様は，平文P，鍵ストリームS，冗長性データ(以下，冗長性と略す)R，初期値Vから暗号文Cを得る暗号処理方法であって，鍵ストリームSの長さが暗号文Cより真に長い。

具体的には，冗長性の長さをbビット，平文の長さを

L=n×b+t (0≦t<b，0≦n)ビットとしたとき，平文の最後にビット文字列`0'を((b-t) mod b)個付加し，

冗長性Rを付加し，全体で

L+((b-t) mod b)+bビットの文字列とする。これはbの倍数である。

これをbビットのブロックに分割し，Pi(1≦i≦m)とする。このとき必要な鍵ストリームの長さは2mbビットである。

この鍵ストリームは，暗号化装置側と復号化装置側とで，秘密裏に事前に共有しておくか，または，事前に共有した秘密鍵(例えば擬似乱数生成器の入力に相当する)から生成する。

上記長さの鍵ストリームを，2つのブロック列，A_i，B_i(1≦i≦m，それぞれbビット)に分割する。

フィードバック値の初期値F_0=Vとして，以下により暗号文ブロックC_iを得る。(この初期値Vも事前に共有しておく。ただし，秘密にしておく必要はない)

$F_i=P_i \hat{} A_i$，　$C_i=(F_i*B_i) \hat{} F_{i-1}$(1≦i≦m)．

得られたC_iをならべ，一つの文字列としたものを暗号文Cとして出力する。ここで，演算子「*」，「^」は，それぞれ有限体$F(2^b)$上での乗算，加算を示す。

【００１２】

対応する復号化は以下のとおりである。

暗号文C'の長さがbビットの倍数でない場合，非受理を出力する。もしbビットの

倍数であったなら，bビットのブロックに分割し，C'_i($1 \leqq i \leqq m'$)とする。

鍵ストリームブロックA_i，B_i($1 \leqq i \leqq m'$)，フィードバック値F'_0=Vをセットして，以下の処理を行う。

$F'_i=(C'_i \hat{} F'_{i-1})/B_i$，　$P'_i=A_i \hat{} F'_i$　($1 \leqq i \leqq m'$).

得られたP'_iをならべ，一つの文字列としたものを復号化結果P'として保存する。演算子「/」は有限体$F(2^b)$上での除算を示す。

bビット文字列P'_mは，改ざんが起こらない場合，冗長性Rが復元されるべきである。また鍵を知らない攻撃者が改ざんを試みて，P'_mが改ざん後にも変化しないような攻撃は$1/2^b$以上で成功しないことが保証されている。よってこれを根拠に，bが十分に大きい(32以上)場合には，P'_mが冗長性Rと同じかどうかを検査することにより改ざんの検出が可能となる。

【００１３】

本態様によれば，暗号文のいずれのブロックに改ざんが施されても，その影響が復号化時に最終ブロックまで伝播するという特徴がある。したがって，冗長データが変化しないような改ざんを，攻撃者が試みても防ぐことが可能になるという効果がある。

より具体的には，暗号化処理において，次のブロックのためのフィードバック値を生成，保存したのち，前のブロックの演算にて生成されたフィードバック値を用いて演算するものである。すなわち，暗号化処理において，t番目ブロックで生成される中間値を，生成される順にX_1,X_2,...,X_nとすると，次のt+1番目ブロックのためのフィードバック値F_tとなるX_iと，前のブロックからのフィードバック値F_{t-1}が作用する中間値X_jとの引数i，jについてi<=jが成り立つ(必要条件)。

本態様によれば，悪意のある改ざんが改ざん検査をパスする確率が$1/2^b$であるが，復号化において有限体での除算が生じ，また，平文の2倍の乱数を消費する。

第二の態様では安全性を少しゆるめて，さらに効率的な処理が可能な方法を示す。

第二の態様でも同様にメッセージ，冗長性の操作を同じように行う。冗長性付き平文がmブロックのとき，必要な鍵ストリームの長さはb×(m+1)ビットであり，こ

れをA_i（$1 \leqq i \leqq m$），Bに分割する。

フィードバック値の初期値F_0=Vとして，以下により暗号文ブロックC_iを得る。

$F_i=P_i \hat{} A_i$，　$C_i=(F_i*B) \hat{} F_{i-1}$ $(1 \leqq i \leqq m)$.

得られたC_iをならべ，一つの文字列としたものを暗号文Cとして出力する。

【００１４】

対応する復号化は以下のとおりである。暗号文C'の長さがbビットの倍数でない場合，非受理を出力する。もしbビットの倍数であったなら，bビットのブロックに分割し，C'_i $(1 \leqq i \leqq m')$とする。暗号化と同様に鍵ストリームブロックA_i（$1 \leqq i \leqq m'$），B，フィードバック値F'_0=Vをセットして，以下の処理を行う。

$F'_i=(C'_i \hat{} F'_{i-1})/B$，　$P'_i=A_i \hat{} F'_i$ $(1 \leqq i \leqq m')$.

得られたP'_iから冗長性の部分を取り出し，決められた冗長性が得られたかどうかを検査する。もしそうであるなら，残りのP'_iの部分をメッセージとして出力し，そうでないなら，非受理を出力する。

ここでbビット文字列P'_mは，改ざんが起こらない場合，冗長性が復元されるべきである。

本態様によれば，i番目ブロックで使う複数の擬似乱数列(鍵ストリーム)のうちいずれかをブロック毎に変化させ，いずれかは共通に用いる。より具体的には，i番目のブロックで，暗号処理に与える疑似乱数列を，A_i，B_iとすると，A_iは毎ブロックごとに変化し，B_iはすべてのブロックを通して変化しない。

この第二の態様では，鍵を知らない攻撃者が改ざん後に改ざん検出で検出されない確率は$(m-1)/2^b$となる。一般的な改ざん検出確率は$1/2^{32}$以下であることが望ましく，mは2^{32}程度が実際の実装でのデータ長の上限であるので，b=64以上が望ましい。この場合，暗号化，復号化では有限体$F(2^{64})$上の乗算を行うことにになる。この演算はハードウェア実装ならば非常に高速かつ低コストで実現可能であるが，ソフトウェア実装の場合，次のような態様による高速化も考えられる。

【００１５】

第三の態様では，より長い冗長性についての方法を示す。まず，以降の処理をbビット単位で行うのを前提として，冗長性をb×dビットとする。第一，第二の態様と同様にメッセージ，冗長性の操作を同じように行い，bビットのメッセージと冗長

性によるブロック列,P_i ($1 \leqq i \leqq m, m \leqq d$)が生成される。そして鍵ストリームの長さをb×(m+d)ビットとし,これをbビットのブロック列A_i ($1 \leqq i \leqq m$) ,B_i ($\neq 0, 1 \leqq i \leqq d$)に分割する。

フィードバック値の初期値$F^{(i)}_0=V_i$($1 \leqq i \leqq d$)として,以下により暗号文ブロックC$_i$を得る。

$F^{(1)}_i=P_i \hat{} A_i$,

$F^{(j+1)}_i=(F^{(j)}_i * B_j) \hat{} F^{(j)}_{i-1}$ ($1 \leqq j \leqq d$),

$C_i=F^{(d+1)}_i$ ($1 \leqq i \leqq m$).

得られたC$_i$をならべ,一つの文字列としたものを暗号文Cとして出力する。

【００１６】

これに対応する復号化は以下のとおりである。暗号文C'の長さがbビットの倍数でない場合,非受理を出力する。もしbビットの倍数であったなら,bビットのブロックに分割し,C'$_i$($1 \leqq i \leqq m'$)とする。暗号化と同様に鍵ストリームブロックA_i ($1 \leqq i \leqq m'$) ,B_i ($\neq 0, 1 \leqq i \leqq d$),フィードバック値$F^{(i)}_0=V_i$($1 \leqq i \leqq d$)をセットして,以下の処理を行う。

$F'^{(d+1)}_i=C'_i$,

$F'^{(j)}_i=(F'^{(j+1)}_i \hat{} F'^{(j)}_{i-1})/B_j$ ($1 \leqq j \leqq d$),

$P'_i=A_i \hat{} F'^{(1)}_i$ ($1 \leqq i \leqq m$).

得られたP'$_i$から冗長性の部分を取り出し,決められた冗長性が得られたかどうかを検査する。もしそうであるなら,残りのP'$_i$の部分をメッセージとして出力し,そうでないなら,非受理を出力する。

【００１７】

第三の態様では,冗長性の長さがb×dビットでありながら,暗号化や復号化に必要な処理は有限体$F(2^b)$上の演算であり,これは有限体$F(2^{b \times d})$上の乗算に比べて一般に$1/d^2$倍となる。しかし,乗算の回数がd倍になることから,この高速化方式は同じ冗長性の長さについて,乗算の演算速度が約1/dの時間で処理可能となる。

第四の態様では,これまで挙げた第一から第三の態様の暗号処理で用いられる有限体$F(2^b)$上の乗算をFeistelの3段構造に組み込んだものである。つまりこれまでのA*Bの演算を,

$M_1=A_L\hat{}(A_R*B_L)$，　$M_2=A_R\hat{}(M_1*B_R)$，　$M_3=M_1\hat{}(M_2*B_L)$，
により$M_3||M_2$を出力する関数に置き換える(B_LとB_R，A_LとA_R，M_2とM_3は，それぞれ入れ替わっても可)。これらの演算は自己逆演算となっており，復号化の際，同じ演算を用いることができる。

【００１８】

第五の態様では，メッセージの分割処理である。平文Pを送信者受信者間の決まりに従ってt個の文字列に分割し，それぞれをP_i($1\leqq i\leqq t$)とする。これまで述べた態様のひとつによりそれぞれ相異なる冗長性R_i($1\leqq i\leqq t$)を用いて暗号化を行い暗号文C_iを得る。またこれとは別にR_iの列のすべて，すなわち，
$(R_1||R_2||R_3||...||R_t)$を平文としてさらに送信者受信者間で共有された冗長性Rを用いて暗号化し，暗号文C_{t+1}を得る。これら，暗号文ブロックの列の連結
$(C_1||C_2||C_3||...||C_{t+1})$を最終的な暗号文Cとする。

【００１９】

対応する復号化は，暗号文を決められた方式に従ってt個の文字列に分割し，それぞれを独立に復号化する。どの復号化結果も非受理でなく，冗長性R_iすべてが，冗長性を平文に見立てた部分の復号化結果に含まれていれば受理として，冗長性の含まれる順番で復号化結果である平文を連結する。どれかの復号化で非受理ならば，非受理とする。

【００２０】

第六の態様では，上記五つの態様について，有限体$F(2^b)$上の乗算のかわりに，有限体F(p)上の乗算に基づいたものを考える(pはある整数kによって2^k+1と表すことができる素数である)。

具体的には$F(2^b)$上の乗算 a*b の代わりとして，F(p)上の演算 a*(b+1)+1 を行う。この演算は2^bビットレジスタ上で，乗算１回，加算２回，シフト1回で処理可能なため，汎用的なプロセッサ上で乗算を高速に行うことができる。

上記演算は，$F(2^b)$の乗算が，排他的論理和b回，シフトb回の計算を要し，一般の素数pにおけるF(p)の乗算が，乗算1回と除算1回(除算は，通常，加算やシフトの数十倍の時間がかかる)を要するのと比較して高速な処理が可能である。

【００２１】

本発明では擬似乱数を用いていることから,ユーザが最も信頼する暗号学的要素アルゴリズムは擬似乱数生成器として，ブロック暗号,ハッシュ関数，ストリーム暗号のどれかを用いることができ,安全性をユーザの選んだ暗号要素アルゴリズムに容易に帰着できる。

実装コストについては,ハッシュ関数の追加実装といった困難な追加実装を避けることができる。

【００２２】

【発明の実施の形態】

(実施例1)

図1はネットワーク10001によって接続されたコンピュータA10002,コンピュータB10003を含む,コンピュータA10002からコンピュータB10003への暗号通信を目的としたシステム構成を示すものである。コンピュータA10002は内部に演算装置(以下CPUという)10004,記憶装置(揮発性，不揮発性を問わない。以下RAMという)10005,ネットワークインターフェース10006を装備しており,外部にはコンピュータA10002をユーザが操作するためのディスプレイ10007とキーボード10008が接続されている。RAM10005には暗号化処理プログラムPROG1_10009,乱数生成処理プログラムPROG2_10010,コンピュータA10002とコンピュータB10003間のみで共有されている秘密情報である秘密鍵K10011,コンピュータA10002とコンピュータB10003間で共有されているデータである冗長性R10012と初期値V10013,それに暗号化してコンピュータB10003に送信したいデータ10014が保存されている。コンピュータB10003は内部にCPU10015,RAM10016,ネットワークインターフェース10017を装備しており,外部にはコンピュータB10003をユーザが操作するためのディスプレイ10018とキーボード10019が接続されている。RAM10016には復号化処理プログラムPROG3_10020,乱数生成処理プログラムPROG2_10021,秘密鍵K10011,冗長性R10012,それに初期値V10013が保存されている。

【００２３】

コンピュータA10002は,暗号化処理プログラムPROG1_10009を実行し,メッセージM10014の暗号文C10022を作成し,ネットワークインターフェース10006を通して

ネットワーク10001へ送信する。コンピュータB10003は，ネットワークインターフェース10017を通して受信したあと，復号化処理プログラムPROG3_10020を実行し，もし改ざんが検出されなければ復号化結果をRAM10016に保存する。

【００２４】

各プログラムは，互いのコンピュータまたは他のコンピュータからネットワーク10001を介して，またはCD,FDなど可搬型記憶媒体を介してRAMに導入することができる。各プログラムは，各コンピュータのオペレーティングシステム(図示していない)の元で動作するように構成することも可能である。

暗号化処理プログラムPROG1_10009は，コンピュータA10002において，RAM10005から読み出されて，CPU10004により実行される。暗号化処理プログラムPROG1_10009は，サブルーチンとして乱数生成処理PROG2_10010を内部で呼び出し，入力秘密鍵K10011，冗長性R10012，初期値V10013，メッセージM10014に対して，暗号文C10022を出力する。

復号化処理プログラムPROG3_10020は，コンピュータB10003において，RAM10016から読み出されてCPU10015により実行される。復号化処理プログラムPROG3_10020は，サブルーチンとして乱数生成処理PROG2_10021を内部で呼び出し，入力秘密鍵K10011，冗長性R10012，初期値V10013，暗号文C10022に対して，メッセージ，または改ざん検知警告を出力する。

【００２５】

暗号化処理プログラムPROG1_10009の処理の流れを説明する。

ステップ20002:データセットサブルーチン。初期値V，冗長性R，秘密鍵Kの入力を待つ。

ステップ20003:平文準備サブルーチン。平文の入力を待ち，平文が与えられたあと決められたパディングを行い，冗長性を付加して，最後に64ビットごとに区切って平文ブロックの列P_i($1 \leqq i \leqq n$)を出力する。

ステップ20004:乱数生成サブルーチン。秘密鍵Kから，擬似乱数列A_i，B_i($1 \leqq i \leqq n$)を出力する。

ステップ20005:暗号化サブルーチン。擬似乱数列A_i，B_i，平文ブロック列P_i($1 \leqq i \leqq n$)，初期値Vを使って，暗号文ブロックC_i($1 \leqq i \leqq n$)を出力する。

ステップ20006:ステップ20005で得られた暗号文ブロックC_i($1 \leqq i \leqq n$)を順にビット連結し,暗号文Cとして出力する。

【００２６】

平文準備サブルーチンの処理を図2を用いて説明する。

【００２７】

ステップ20202:暗号処理に用いるメッセージMの入力を待つ。メッセージMは,キーボード10008から入力されたり,RAM内に保存されていたり,他の記憶媒体から導入されたりする。

ステップ20203:メッセージの長さを示すデータをパディング。メッセージMの先頭にメッセージMのビット長を表す64ビット二進数データを付加する。

ステップ20204:メッセージ長をそろえるパディング。後の暗号処理のためパディング後のデータを64ビットの整数倍にする。メッセージMの長さをLビットとすると$64-L \pmod{64}$個の0をステップ20203で長さデータを付加されたメッセージの末尾にパディングする。

ステップ20205:冗長性データのパディング。64ビットデータとして与えられる冗長性Rをさらに末尾にパディングする。

ステップ20206:メッセージデータの平文ブロックへ分割。ステップ20205の結果得られたデータを64ビットのブロックに区切り順に$P_1, P_2, \cdots, P_n$とする。

【００２８】

乱数生成サブルーチンの処理を図3を用いて説明する。

ステップ20302:必要パラメータの入力。パディング後のメッセージブロック数nと,秘密鍵Kを得る。

ステップ20303:擬似乱数列Aの生成。乱数生成処理プログラムPROG2を呼び出し,長さ64nビットの擬似乱数列を生成,出力をAとする。

ステップ20304:乱数列Aをブロックに分割。擬似乱数列Aを64ビットごとに区切り順に$A_1, A_2, \cdots, A_n$とする。

ステップ20305:カウンタiの初期化。カウンタi=1とする。

ステップ20306:乱数B_iの生成。秘密鍵Kを使ってPROG2を実行することにより64ビットの乱数B_iを生成する。

ステップ20307:ステップ20306で生成されたB_iが0ならば,ステップ20306に戻る。

ステップ20308:i=nならばステップ20310を実行。

ステップ20309:カウンタiをインクリメントしステップ20306へ戻る。

【００２９】

暗号化サブルーチンの処理を図4を用いて説明する。

ステップ20402:初期値セット。F_0=Vとする。

ステップ20403:カウンタ初期化。i=1とする。

ステップ20404:フィードバック値F_i計算。$F_i=P_i \hat{} A_i$とする。

ステップ20405:暗号文ブロックC_i計算。$C_i=(F_i*B_i) \hat{} F_{i-1}$とする。

ステップ20406:i=nならばステップ20408を実行。

ステップ20407:カウンタiをインクリメントしステップ20404へ戻る。

【００３０】

復号化処理プログラムPROG3_10020の処理の流れを図5を用いて説明する。

ステップ20502:データセットサブルーチン。初期値V,冗長性R,秘密鍵Kの入力を待つ。

ステップ20503:暗号文準備サブルーチン。暗号文C'の入力を待ち,暗号文C'が与えられたあと64ビットごとに区切って暗号文ブロックの列C'_i($1 \leqq i \leqq n$)を出力する。

ステップ20504:乱数生成サブルーチン。秘密鍵Kから,擬似乱数列A_i,B_i($1 \leqq i \leqq n$)を出力する。

ステップ20505:復号化サブルーチン。擬似乱数列A_i,B_i,暗号文ブロック列C'_i($1 \leqq i \leqq n$),初期値Vを使って,平文ブロックP'_i($1 \leqq i \leqq n$)を出力する。

ステップ20506:平文切り出しサブルーチン。平文ブロックの列P'_iを3つのデータ列,L',M',Z'に分割する。

ステップ20507:冗長性切り出しサブルーチン。Z'をR'とT'に分割する。

ステップ20508:T=0かつR'=Rならばステップ20510に進む。

ステップ20509:拒否(非受理)を出力。ステップ20511に進む。

ステップ20510:M'をRAMへ格納する。

ステップ20509,20510では，復号化処理プログラムは，結果(受理/非受理あるいは復号結果)をディスプレイ10018に出力して，ユーザに結果を通知する。

【００３１】

暗号文準備サブルーチンの処理を図6を用いて説明する。

ステップ20602:暗号文C'の入力を待つ。

ステップ20603:暗号文C'を64ビットごとに区切り,順にC'_1,C'_2,…,C'_nとする

復号化サブルーチンの処理を図7を用いて説明する。

【００３２】

ステップ20702:初期値セット。F'_0=Vとする。

ステップ20703:カウンタ初期化。i=1とする。

ステップ20704:フィードバック値F'_i計算。$F'_i=(C'_i \hat{} F'_{i-1})/B_i$とする。

ステップ20705:平文ブロックP'i計算。$P'_i=F'_i \hat{} A_i$とする。

ステップ20706:i=nならばステップ20708を実行。

ステップ20707:カウンタiをインクリメントしステップ20704へ戻る。

【００３３】

平文切り出しサブルーチンの処理を図8を用いて説明する。

ステップ20802:L'を最初の64ビットの平文ブロックとする。

ステップ20803:復号文ブロックのうちP'_2最上位ビットからL'ビットのデータをM'とする。

ステップ20804:M'を切り取った残りのデータをZ'とする。

【００３４】

冗長性切出しサブルーチンの処理を図9を用いて説明する。

ステップ20902:R'をZ'の下位64ビットのデータとする。

ステップ20903:R'を切り取った後の残りのデータをT'とする。

【００３５】

図10は，暗号化処理の説明図である。○印の中に+が入った記号は64ビット幅の二つのデータによる排他的論理和演算を表し,○印の中に×が入った記号は64ビット幅の二つのデータによる有限体$F(2^{64})$上での乗算を示す。

メッセージM20931に長さ20930,適当なパディング20932,冗長性R20933をそれぞ

れ付加し，平文P20934を生成する。

これを64ビットに分割したものをそれぞれP_1_20935，P_2_20936，P_3_20937，P_n_20938とする。

P_1_20935はA_1_20940と排他論理和を取り，フィードバック値F_1_20941とし，B_1_20942と有限体上の乗算を行い，その結果と初期値F_0_20939の排他論理和により暗号文ブロックC_1_20943を得る。

P_2_20936はA_2_20946と排他論理和を取り，フィードバック値F_2_20945とし，B_2_20946と有限体上の乗算を行い，その結果とフィードバック値F_1_20941の排他論理和により暗号文ブロックC_2_20947を得る。

同様にしてP_n_20938まで処理を行い，暗号文ブロックC_1_20943，C_2_20947，C_3_20951，...，C_n_20955を得たあと，これらをこの順で連結し，暗号文C_20956を得る。

【００３６】

図11は，復号化処理の説明図である。○印の中に/が入った記号は64ビット幅の二つのデータによる有限体$F(2^{64})$上での除算を示す。該記号の上から入るデータが割られる数，該記号の左から入るデータが割る数である。

暗号文C'_20960を64ビットのブロックに分割し，C'_1_20962，C'_2_20963，C'_3_20964，...，C'_n_20965とする。

C'_1は初期値F'_0_20961と排他的論理和をとり，その結果をB_1_20966で割る。この結果をフィードバック値F'_1_20967とする。平文ブロックP'_1_20969はフィードバック値F'_1_20967とA_1_20968との排他論理和により得られる。

C'_2_20963，C'_3_20964，...，C'_n_20965も同様に処理を行い，得られた平文ブロックP'_1_20969，P'_2_20972，P'_3_20977，...，P'_n_20981を連結し平文P'_20982とする。これをL'_20983，M'_20984，Z'_209985に分割後，さらにZ'_209985をT'_20988とR'_20989に分割し，冗長性のチェックを行う。

【００３７】

実施例1では，メッセージの長さの約2倍の長さだけの擬似乱数を暗号処理に使った。擬似乱数はブロック暗号に比較して高速ではあるものの，この暗号処理で最も計算量のかかる処理であり，消費する乱数の数を減らすことが望ましい。

(実施例2)

以下に示す実施例2では，別の関数を使うことにより，より少ない擬似乱数の消費で行え，また復号化の際の割る数も変化せず，逆数の事前計算によりほぼ乗算と同じ速度で達成できるため，極めて効率的な処理が可能となる。

実施例2では，暗号化処理PROG1，復号化処理PROG3の代わりに暗号化処理PROG1A，復号化処理PROG3Aをそれぞれ使う。

暗号化処理PROG1Aを示す。PROG1Aは，図1のPROG1_10009において，乱数生成サブルーチン20004と暗号化サブルーチン20005をそれぞれ乱数生成2サブルーチン21004と暗号化2サブルーチン21005に置き換えたものである。

【００３８】

乱数生成2サブルーチン21004の処理を図12を用いて説明する。

【００３９】

ステップ21102：必要パラメータの入力。パディング後のメッセージブロック数nと，秘密鍵Kを得る。

ステップ21103：擬似乱数列Aの生成。乱数生成処理プログラムPROG2を呼び出し，長さ64nビットの擬似乱数列を生成，出力をAとする。

ステップ21104：乱数列Aをブロックに分割。擬似乱数列Aを64ビットごとに区切り順にA_1，A_2，…，A_nとする。

ステップ21105：乱数Bの生成。秘密鍵Kを使ってPROG2を実行することにより64ビットの乱数Bを生成する。

ステップ21106：ステップ21105で生成されたBが0ならば，ステップ21105に戻る。

【００４０】

暗号化2サブルーチン21005の処理を図13を用いて説明する。

ステップ21202：初期値セット。F_0=Vとする。

ステップ21203：カウンタ初期化。i=1とする。

ステップ21204：フィードバック値F_i計算。F_i=P_i＾A_iとする。

ステップ21205：暗号文ブロックC_i計算。C_i=(F_i*B)＾F_{i-1}とする。

ステップ21206：もし，i=nならばステップ21208を実行。

ステップ21207：カウンタiをインクリメントしステップ21204へ戻る。

【００４１】

PROG1Aと対となる,復号化処理PROG3Aの処理の流れを図14を用いて説明する。PROG3Aは,PROG3_10020において,乱数生成サブルーチン20504と復号化サブルーチン20505をそれぞれ乱数生成2サブルーチン21304と復号化2サブルーチン21305に置き換えたものである。

【００４２】

ステップ21302:データセットサブルーチン。初期値V,冗長性R,秘密鍵Kの入力を待つ。

ステップ21303:暗号文準備サブルーチン。暗号文C'の入力を待ち,暗号文C'が与えられたあと64ビットごとに区切って暗号文ブロックの列C'_i($1 \leqq i \leqq n$)を出力する。

ステップ21304:乱数生成サブルーチン。秘密鍵Kから,擬似乱数列A_i($1 \leqq i \leqq n$),Bを出力する。

ステップ21305: 復号化サブルーチン。擬似乱数列A_i,B,暗号文ブロック列C'_i($1 \leqq i \leqq n$),初期値Vを使って,平文ブロックP'_i($1 \leqq i \leqq n$)を出力する。

ステップ21306:平文切り出しサブルーチン。平文ブロックの列P'_iを3つのデータ列,L',M',Z'に分割する。

ステップ21307:冗長性切り出しサブルーチン。Z'をR'とT'に分割する。

ステップ21308:T=0かつR'=Rならばステップ21310に進む。

ステップ21309:拒否を出力。ステップ21311に進む。

ステップ21310:M'をRAMへ格納する。

【００４３】

図14の復号化2サブルーチン21305の処理を図15を用いて説明する。

ステップ21402:初期値セット。F'_0=Vとする。

ステップ21403:1/Bを事前計算しておく。

ステップ21404:カウンタ初期化。i=1とする。

ステップ21405:フィードバック値F'_i計算。$F'_i=(C'_i \hat{} F'_{i-1})*(1/B)$とする。

ステップ21406:平文ブロックP'_i計算。$P'_i=F'_i \hat{} A_i$とする。

ステップ21407:i=nならばステップ21409を実行。

ステップ21408:カウンタiをインクリメントしステップ21405へ戻る。

【００４４】

図16は，上述の高速化手法で用いた暗号化処理の説明図である。

メッセージM21421に長さ21420,適当なパディング21422,冗長性R21423をそれぞれ付加し,平文P21424を生成する。

これを64ビットに分割したものをそれぞれP_1_21425,P_2_21426,P_3_21427,P_n_21428とする。

P_1_21425はA_1_21431と排他論理和を取り,フィードバック値F_1_21432とし，B_21429と有限体上の乗算を行い,その結果と初期値F_0_21430の排他論理和により暗号文ブロックC_1_21433を得る。

P_2_21426はA_2_21434と排他論理和を取り,フィードバック値F_2_21435とし，B_21429と有限体上の乗算を行い,その結果とフィードバック値F_1_21432の排他論理和により暗号文ブロックC_2_21436を得る。

同様にしてP_n_21428まで処理を行い,暗号文ブロックC_1_21433,C_2_21436,C_3_21439,...,C_n_21442を得たあと,これらをこの順で連結し,暗号文C_21443を得る。

【００４５】

図17は,対応する復号化処理の説明図である。

暗号文C'_21450は64ビットのブロックに分割し,C'$_1$_21453,C'$_2$_21454,C'$_3$_21455,...,C'$_n$_212456とする。

C'1は初期値F'$_0$_21451と排他的論理和をとり,その結果と1/B_21452の乗算を計算する。この結果をフィードバック値F'$_1$_21457とする。平文ブロックP'$_1$_21459はフィードバック値F'$_1$_21457とA_1_21458との排他論理和により得られる。

C'$_2$_21454,C'$_3$_21455,...,C'$_n$_21456も同様に処理を行い,得られた平文ブロックP'$_1$_21459,P'$_2$_21462,P'$_3$_21465,...,P'$_n$_21468を連結し平文P'_21476とする。これをL'_21469,M'_21470,Z'_21471に分割後,さらにZ'_21471をT'_21474とR'_21475に分割し,冗長性のチェックを行う。

実施例2では,64ビットの冗長性を用いるために有限体F(2^{64})上の加算,乗算を用いている。

(実施例3)

以下に示す実施例3では，別の高速な関数を使うことにより同じ安全性かつより高速な処理が可能となる。観点を変え，これまでと同じF(2^{64})上の演算を使った場合には(F(2^{128})の演算相当の)より高い安全性を提供することができる。

高速化の観点では，一般にF(2^{64})上の乗算はF(2^{32})上のそれに比べ4倍の計算量がかかるが,実施例3では1/4の計算量の演算を2回繰り返すので事実上2倍の高速処理が可能となる。

安全性の補強という観点では，F(2^{64})上の演算と64ビットのフィードバック値を2度用いることで，改ざんの成功確率を，上記手法の2^{-64}から2^{-128}へより小さくすることができる。

この高速化手法として暗号化処理PROG1の代わりに同じく暗号化処理PROG1B,復号化処理PROG3の代わりに同じく復号化処理PROG3Bをそれぞれ使う装置を考える。

【００４６】

暗号化処理PROG1Bは,図1に示すPROG1_1009において,乱数生成サブルーチン(ステップ20004)と暗号化サブルーチン(ステップ20005)をそれぞれ乱数生成3サブルーチン21504と暗号化3サブルーチン21505に置き換えたものである。暗号化処理PROG1Bの処理の流れを図18を用いて説明する。

ステップ21502:データセットサブルーチン。初期値V,冗長性R,秘密鍵Kの入力を待つ。

ステップ21503:平文準備サブルーチン。平文の入力を待ち,平文が与えられたあと決められたパディングを行い,冗長性を付加して,最後に32ビットごとに区切って平文ブロックの列P_i($1 \leqq i \leqq n$)を出力する。

ステップ21504:乱数生成3サブルーチン。秘密鍵Kから,擬似乱数列A_i($1 \leqq i \leqq n$),Ba,Bbを出力する。

ステップ21505:暗号化3サブルーチン。擬似乱数列A_i,Ba,Bb,平文ブロック列P_i($1 \leqq i \leqq n$),初期値Vを使って,暗号文ブロックC_i($1 \leqq i \leqq n$)を出力する。

ステップ21506:ステップ21505で得られた暗号文ブロックC_i($1 \leqq i \leqq n$)を順にビット連結し,暗号文Cとして出力する。

【００４７】

乱数生成3サブルーチン21504の処理を図19を用いて説明する。

ステップ21602:必要パラメータの入力。パディング後のメッセージブロック数nと,秘密鍵Kを得る。

ステップ21603:擬似乱数列Aの生成。乱数生成処理プログラムPROG2を呼び出し,長さ32nビットの擬似乱数列を生成,出力をAとする。

ステップ21604:乱数列Aをブロックに分割。擬似乱数列Aを32ビットごとに区切り順にA_1,A_2,…,A_nとする。

ステップ21605:乱数Baの生成。秘密鍵Kを使ってPROG2を実行することにより32ビットの乱数Baを生成する。

ステップ21606:ステップ21605で生成されたBaが0ならば,ステップ21605に戻る。

ステップ21607:乱数Bbの生成。秘密鍵Kを使ってPROG2を実行することにより32ビットの乱数Bbを生成する。

ステップ21608:ステップ21607で生成されたBbが0ならば,ステップ21607に戻る。

【００４８】

暗号化3サブルーチン21505の処理を図20を用いて説明する。記号「*」,「^」はそれぞれ有限体$F(2^{32})$上での乗算,加算を表す。

ステップ21702:初期値セット。$FA_0=FB_0=V$とする。

ステップ21703:カウンタ初期化。i=1とする。

ステップ21704:フィードバック値FA_i計算。$FA_i=P_i \hat{} A_i$とする。

ステップ21705:フィードバック値FB_i計算。$FB_i=(FA_i*Ba) \hat{} FA_{i-1}$とする。

ステップ21706:暗号文ブロックC_i計算。$C_i=(FB_i*Bb) \hat{} FB_{i-1}$とする。

ステップ21707:i=nならばステップ21709を実行。

ステップ21708:カウンタiをインクリメントしステップ21704へ戻る。

【００４９】

復号化処理PROG3Bの処理の流れを図21を用いて説明する。PROG3Bは,PROG3_10020において，乱数生成サブルーチン20504と復号化サブルーチン20505をそれぞれ

乱数生成3サブルーチン21804と復号化3サブルーチン21805に置き換えたものである。

ステップ21802:データセットサブルーチン。初期値V,冗長性R,秘密鍵Kの入力を待つ。

ステップ21803: 暗号文準備サブルーチン。暗号文C'の入力を待ち,暗号文C'が与えられたあと32ビットごとに区切って暗号文ブロックの列C'_i($1 \leqq i \leqq n$)を出力する。

ステップ21804:乱数生成サブルーチン。秘密鍵Kから,擬似乱数列A_i($1 \leqq i \leqq n$),Ba,Bbを出力する。

ステップ21805:復号化サブルーチン。擬似乱数列A_i,Ba,Bb,暗号文ブロック列C'_i($1 \leqq i \leqq n$),初期値Vを使って,平文ブロックP'_i($1 \leqq i \leqq n$)を出力する。

ステップ21806:平文切り出しサブルーチン。平文ブロックの列P'_iを3つのデータ列,L',M',Z'に分割する。

ステップ21807:冗長性切り出しサブルーチン。Z'をR'とT'に分割する。

ステップ21808:T=0かつR'=Rならばステップ21810に進む。

ステップ21809:拒否を出力。ステップ21811に進む。

ステップ21810:M'をRAMへ格納する。

【００５０】

図21の復号化3サブルーチン21805の処理を図22を用いて説明する。記号「/」は有限体$F(2^{32})$上での除算を表す。

ステップ21902:初期値セット。$FA'_0=FB'_0=V$とする。

ステップ21903:1/Ba,1/Bbを事前計算しておく。

ステップ21904:カウンタ初期化。i=1とする。

ステップ21905:フィードバック値FB'_i計算。$FB'_i=(C'_i \hat{} FB'_{i-1})*(1/Bb)$とする。

ステップ21906:フィードバック値FA'_i計算。$FA'_i=(FB'_i \hat{} FA'_{i-1})*(1/Ba)$とする。

ステップ21907:平文ブロックP'_i計算。$P'_i=FA'_i \hat{} A_i$とする。

ステップ21908:i=nならばステップ21910を実行。

ステップ21909:カウンタiをインクリメントしステップ21905へ戻る。

【００５１】

図23は，上述の高速化手法で用いた暗号化処理の説明図である。

メッセージM21921に長さL21920,適当なパディング21922,冗長性R21923をそれぞれ付加し,平文P21924を生成する。

これを64ビットに分割したものをそれぞれP_1_21925,P_2_21926,P_3_21927,P_n_21928とする。

P_1_21925はA_1_21933と排他論理和を取り,フィードバック値FA_1_21934とし，Ba_21929と有限体上の乗算を行い,その結果と初期値FA_0_21930の排他論理和により,フィードバック値FB_1_21939を得る。これをBb_21931と有限体上の乗算を行い,その結果と初期値FB_0_21932の排他論理和により暗号文ブロックC_1_21936を得る。

P_2_21926はA_2_21937と排他論理和を取り,フィードバック値FA_2_21942とし，Ba_21929と有限体上の乗算を行い,その結果とフィードバック値FA_1_21934の排他論理和により,フィードバック値FB_2_21943を得る。これをBb_21931と有限体上の乗算を行い,その結果とフィードバック値FB_1_21939の排他論理和により暗号文ブロックC_2_21940を得る。

同様にしてP_n_21928まで処理を行い,暗号文ブロックC_1_21936,C_2_21940,C_3_21944,...,C_n_21950を得たあと,これらをこの順で連結し,暗号文C_21951を得る。

【００５２】

図24は，対応する復号化処理の説明図である。

暗号文C'_21960は64ビットのブロックに分割し,C'_1_21961,C'_2_21962,C'_3_21963,...,C'_n_21964とする。

C'_1は初期値FB'_0_21965と排他的論理和をとり,その結果と1/Bb_21966の乗算を計算する。この結果をフィードバック値FB'_1_21969とする。さらに,初期値FA'_0_21967と排他的論理和をとり,その結果と1/Ba_21968の乗算を計算する。平文ブロックP'_1_21972はフィードバック値FA'_1_21970とA_1_21971との排他論理和により得られる。

C'_2_21962,C'_3_21976,...,C'_n_21964も同様に処理を行い,得られた平文ブロックP'_1_21972,P'_2_21976,P'_3_21980,...,P'_n_21985を連結し平文P'_21986とする。

これをL'_21987,M'_21988,Z'_21989に分割後，さらにZ'_21989をT'_21992とR'_21993に分割し，冗長性のチェックを行う。

【００５３】

(実施例4)

上記実施例1から3は並列計算を考えない単一プロセッサによる処理の実施形態であった。実施例4において，本発明が並列処理への応用も簡単であることを示す。

実施例4のシステム構成が，図１の構成と異なるのは，コンピュータA10002については，CPU10004の代わりにCPU1_30004,CPU2_30005を装備しており,RAM10005には，図１の構成に加えて並列暗号処理プログラムPROG4_30016が保存されている点である。コンピュータB10003については，CPU10015の代わりにCPU1_30017,CPU2_30018を装備しており,RAM10016には，図１の構成に加えて並列復号処理プログラムPROG5_30025が保存されている点が異なる。

コンピュータA10002は,並列暗号化処理プログラムPROG4_30016を実行し,メッセージM10014の暗号文C10022を作成し,送信する。コンピュータB10003は,暗号文C10022を受信したあと,並列復号化処理プログラムPROG5_30025を実行し,もし改ざんが検出されなければ復号化結果をRAM10016に保存する。

並列暗号化処理プログラムPROG4_30016は,コンピュータA10002において，RAM10005から読み出されて,CPU1_30004とCPU2_30005により実行されることで実現される。並列暗号化処理プログラムPROG4_30016は，サブルーチンとして暗号化処理プログラムPROG1_10009,乱数生成処理PROG2_10010を内部で呼び出し,実行する。

並列復号化処理プログラムPROG5_30025は,コンピュータB10003において，RAM10016から読み出されてCPU1_30017とCPU2_30018により実行される。並列復号化処理プログラムPROG5_30025は,サブルーチンとして復号化処理プログラムPROG3_10020,乱数生成処理PROG2_10021を内部で呼び出し,実行する。

その他の構成，動作は図１と同じである。

【００５４】

並列暗号化処理プログラムPROG4_30016の処理の流れを図25を用いて説明する。記号「A||B」はビット列AとBの連結を示す。

ステップ40002:CPU1の実行するメッセージ処理によりメッセージMをM_1,M_2に2分割する。

ステップ40003:CPU1が実行するPROG1_10009による暗号化処理により,初期値V+1,冗長性R+1,秘密鍵K,平文M_1,を使って,暗号文C_1を出力する。

ステップ40004:CPU2が実行するPROG1_10009による暗号化処理により,初期値V+2,冗長性R+2,秘密鍵K,平文M_2,を使って,暗号文C_2を出力する。

ステップ40005:CPU1が実行するPROG1_10009による暗号化処理により,初期値V,冗長性R,秘密鍵K,平文$(R_1||R_2)$,を使って,暗号文C_3を出力する。

ステップ40006:暗号文Cの生成。$C=C_1||C_2||C_3$とする。

ステップ40007:暗号文Cをメモリへ保存

並列復号化処理プログラムPROG5_30025の処理の流れを図26を用いて説明する。

ステップ40102:CPU1の実行する暗号文処理により暗号文C'をC'_1,C'_2,C'_3に3分割する。C'_3は192ビットであり,C'_1,C'_2は同じ長さ,かつ$C'=C'_1||C'_2||C'_3$である。

ステップ40103:CPU1が実行するPROG3_10020による復号化処理により,初期値V+1,冗長性R+1,秘密鍵K,暗号文ブロックC'_1,を使って,復号化を行い結果を保存する。

ステップ40104:CPU2が実行するPROG3_10020による復号化処理により,初期値V+2,冗長性R+2,秘密鍵K,暗号文ブロックC'_2を使って,復号化を行い,結果を保存する。

ステップ40105:ステップ40103の復号化結果,ステップ40104の復号化結果のうち少なくともひとつが非受理ならばステップ40111を実行する。

ステップ40106: CPU1が実行するPROG3_10020による復号化処理により,初期値V,冗長性R,秘密鍵K,暗号文ブロックC'_3を使って,復号化を行い,結果を保存する。

ステップ40107:ステップ40106の復号化結果が非受理ならばステップ40111を実行する。

ステップ40108:ステップ40106の復号化結果が(R+1)||(R+2)でないなら,ステップ40111を実行する。

ステップ40109:復号化結果$M'=M'_1||M'_2$とする。

ステップ40110:M'をメモリへ格納し,ステップ40112を実行する。

ステップ40111:非受理を出力する。

【００５５】

実施例4により独立な二つのプロセッサを使った並列暗号処理が可能となる。

図27は，上述の並列化手法で用いた暗号化処理の説明図である。

メッセージM401402を分割して生成したM_1_40141,M_2_40142にそれぞれ冗長性,R+1,R+2を付加した,40143,40144を暗号化処理40146,40147により暗号化,暗号文ブロックC_1_40149,C_2_40150を得る。

さらに冗長性R+1,R+2をメッセージとして冗長性をR使って暗号化し,暗号文ブロックC_3_40151を得る。

暗号文ブロックC_1_40149,C_2_40150,C_3_40151を連結し暗号文C40152として出力する。

【００５６】

図28は,対応する並列復号化処理の説明図である。

暗号文C'40160を3分割し,C'_1_40161,C'_2_40162,C'_3_40163を生成する。

それぞれを復号化処理40164,40165,40166により復号化,平文ブロック40167,40168,40169を得る。

これらすべてが受理であって,平文ブロック40167,40168の冗長性が,平文ブロック40169のメッセージ部と同じであり,さらに平文ブロック40169が予め共有されている冗長性であれば,平文ブロック40167,40168からそれぞれメッセージ部M'_1_40170,M'_2_40171を取り出し,これらを連結してメッセージM'40172を得る。

【００５７】

以上の各実施例におけるCPUはプログラムを実行するものであれば，汎用，専用を問わず用いることができる。また，各実施例は，CPUがプログラムを実行することにより実現される例であったが，各処理を専用のハードウェアを用いても，高速にかつ，低価格で実現することも可能である。

【００５８】

【発明の効果】

より安全で高速な共通鍵暗号技術を提供することが可能となる。

【図面の簡単な説明】

【図１】

各実施例のシステム構成図を示す。

【図２】

平文準備サブルーチンのフロー図を示す。

【図３】

乱数生成サブルーチンのフロー図を示す。

【図４】

暗号化サブルーチンのフロー図を示す。

【図５】

図1の復号化処理プログラムのフロー図を示す。

【図６】

図7の暗号文準備サブルーチンのフロー図を示す。

【図７】

図7の復号化サブルーチンのフロー図を示す。

【図８】

図7の平文切出しサブルーチンのフロー図を示す。

【図９】

図7の冗長性切出しサブルーチンのフロー図を示す。

【図１０】

暗号化処理の,データブロックによる図を示す。

【図１１】

図7で示した復号化処理の,データブロックによる図を示す。

【図１２】

実施例2の乱数生成2サブルーチンのフロー図を示す。

【図１３】

実施例2の暗号化2サブルーチンのフロー図を示す。

【図１４】

実施例2の復号化処理プログラムのフロー図を示す。

【図１５】

実施例2の復号化2サブルーチンのフロー図を示す。

【図１６】

実施例2の暗号化処理の,データブロックによる図を示す。

【図１７】

実施例2の復号化処理の,データブロックによる図を示す。

【図１８】

実施例3の暗号化処理プログラムのフロー図を示す。

【図１９】

実施例3の乱数生成3サブルーチンのフロー図を示す。

【図２０】

実施例3の暗号化3サブルーチンのフロー図を示す。

【図２１】

実施例3の復号化処理プログラムのフロー図を示す。

【図２２】

実施例3の復号化3サブルーチンのフロー図を示す。

【図２３】

実施例3の暗号化処理の,データブロックによる図を示す。

【図２４】

実施例3の復号化処理の,データブロックによる図を示す。

【図２５】

実施例4の並列暗号化処理プログラムのフロー図を示す。

【図２６】

実施例4の並列復号化処理プログラムのフロー図を示す。

【図２７】

実施例4の暗号化処理の,データブロックによる図を示す。

【図２８】

実施例4の復号化処理の,データブロックによる図を示す。

【符号の説明】

10001,30001:ネットワーク

10002,10003,30002,30003:コンピュータ

10004,10015,30004,40005,30017,30018:CPU

10005,10016,30006,30019:メモリ

10006,10017,30007,30020:ネットワークインターフェース

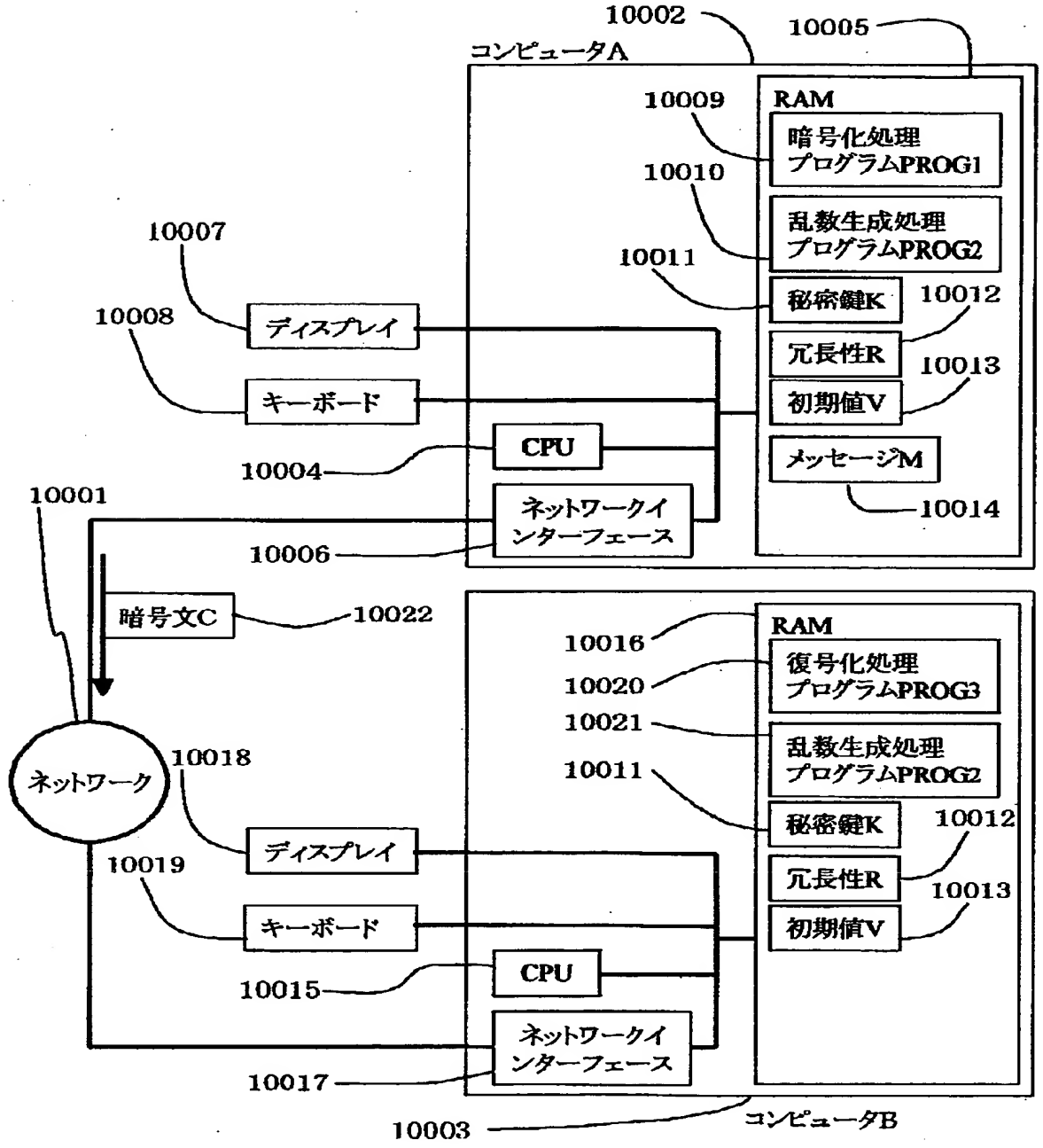
10007,10018,30008,30021:ディスプレイ

10008,10019,30009,30022:キーボード

【書類名】　図面

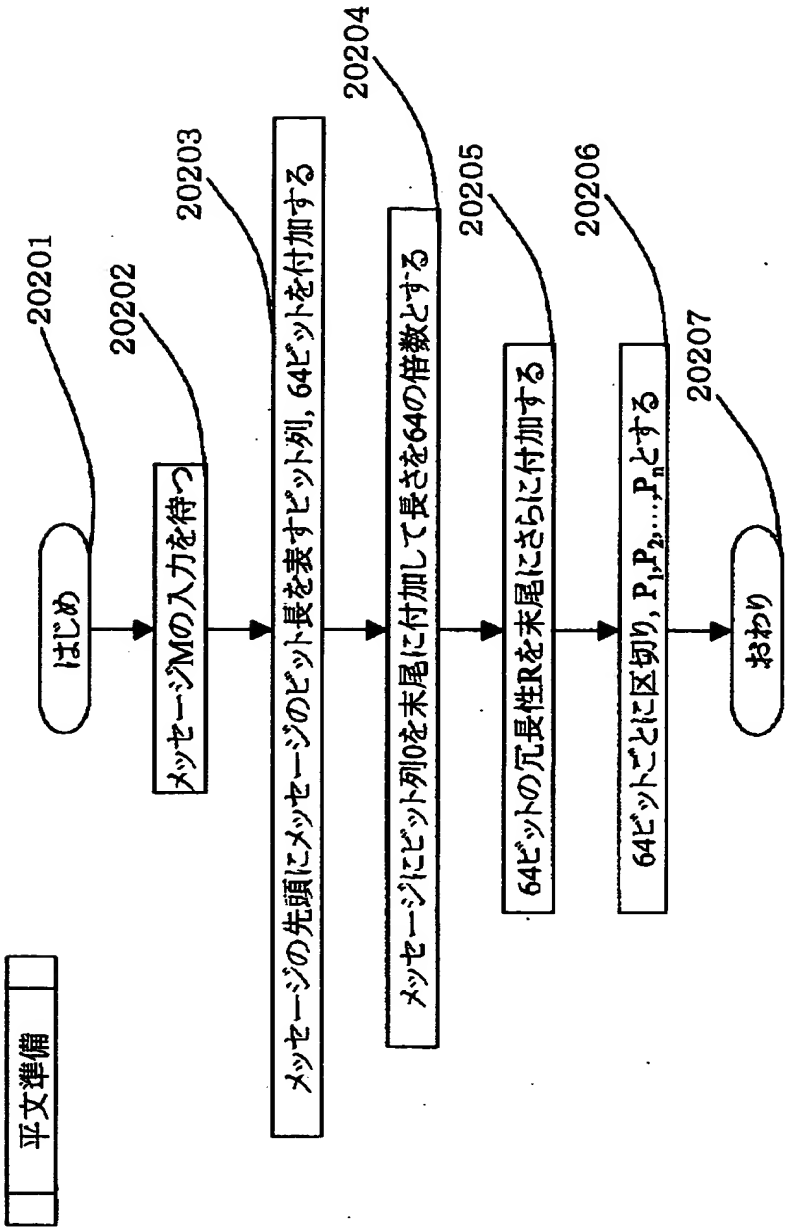
【図１】

図1


10002
10005
コンピュータA
10009
RAM
暗号化処理
プログラムPROG1
10010
乱数生成処理
プログラムPROG2
10007
10011
秘密鍵K
10012
10008
ディスプレイ
冗長性R
10013
キーボード
初期値V
CPU
メッセージM
10004
10014
10001
ネットワークイ
ンターフェース
10006
暗号文C
10022
10016
RAM
復号化処理
プログラムPROG3
10020
10021
乱数生成処理
プログラムPROG2
10011
ネットワーク
10018
秘密鍵K
10012
ディスプレイ
10019
冗長性R
10013
キーボード
初期値V
CPU
10015
ネットワークイ
ンターフェース
10017
10003
コンピュータB

【図２】

図2


平文準備
はじめ
20201
メッセージMの入力を待つ
20202
メッセージの先頭にメッセージのビット長を表すビット列，64ビットを付加する
20203
メッセージにビット列0を末尾に付加して長さを64の倍数とする
20204
64ビットの冗長性Rを末尾にさらに付加する
20205
64ビットごとに区切り，$P_1,P_2,\ldots,P_n$とする
20206
おわり
20207

【図３】

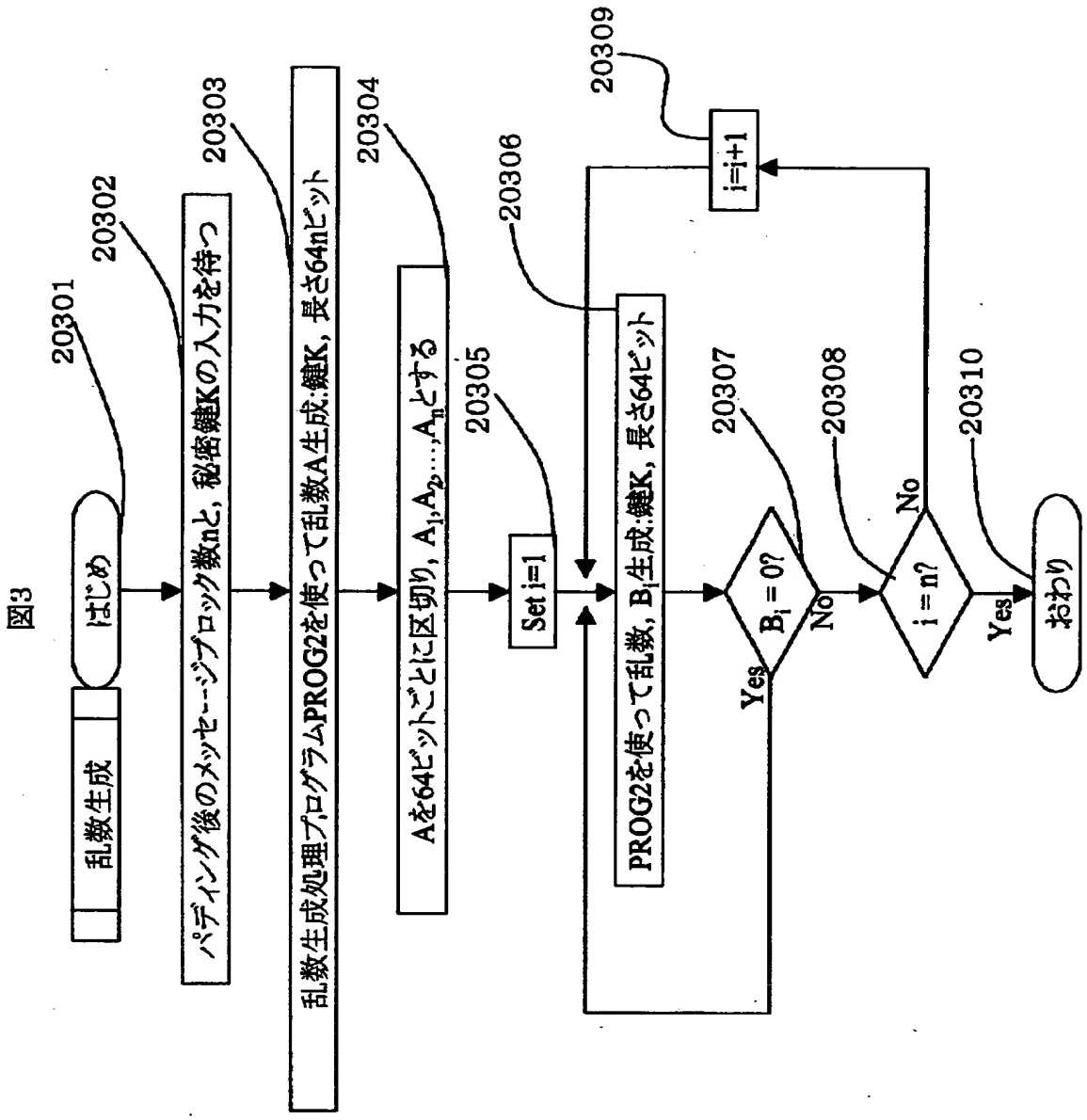
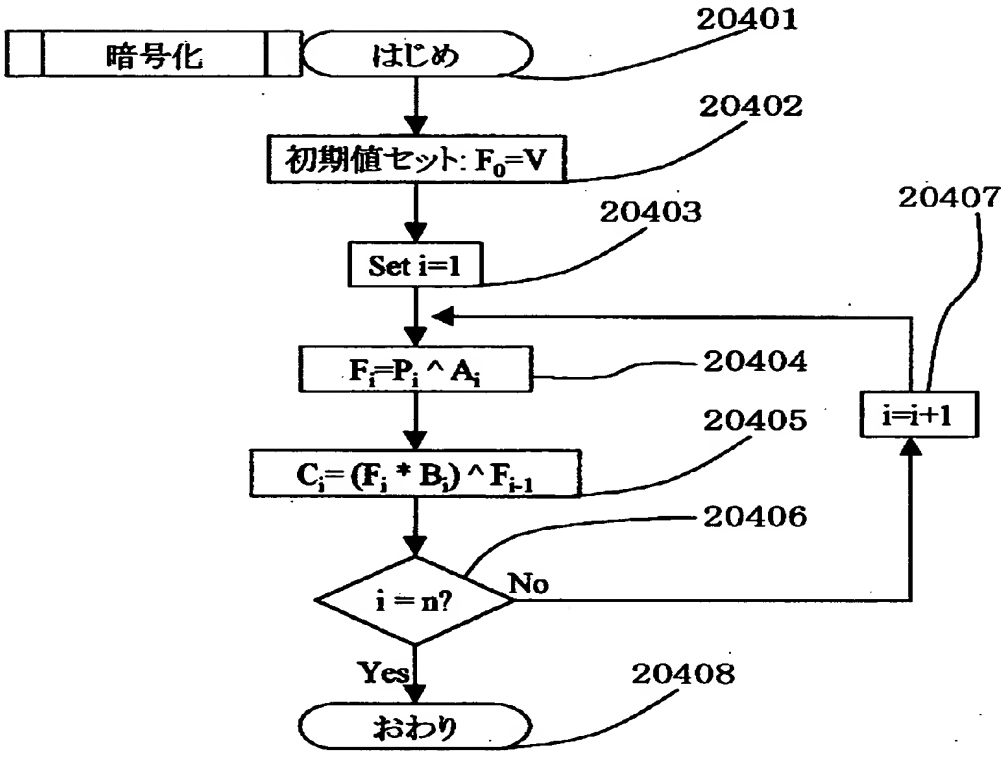


図3
はじめ
乱数生成
20301
パディング後のメッセージブロック数nと，秘密鍵Kの入力を待つ
20302
乱数生成処理プログラムPROG2を使って乱数A生成:鍵K，長さ64nビット
20303
Aを64ビットごとに区切り，$A_1,A_2,\ldots,A_n$とする
20304
Set i=1
20305
PROG2を使って乱数，B_i生成:鍵K，長さ64ビット
20306
$B_i = 0$?
20307
Yes
No
i = n?
20308
No
Yes
i=i+1
20309
おわり
20310

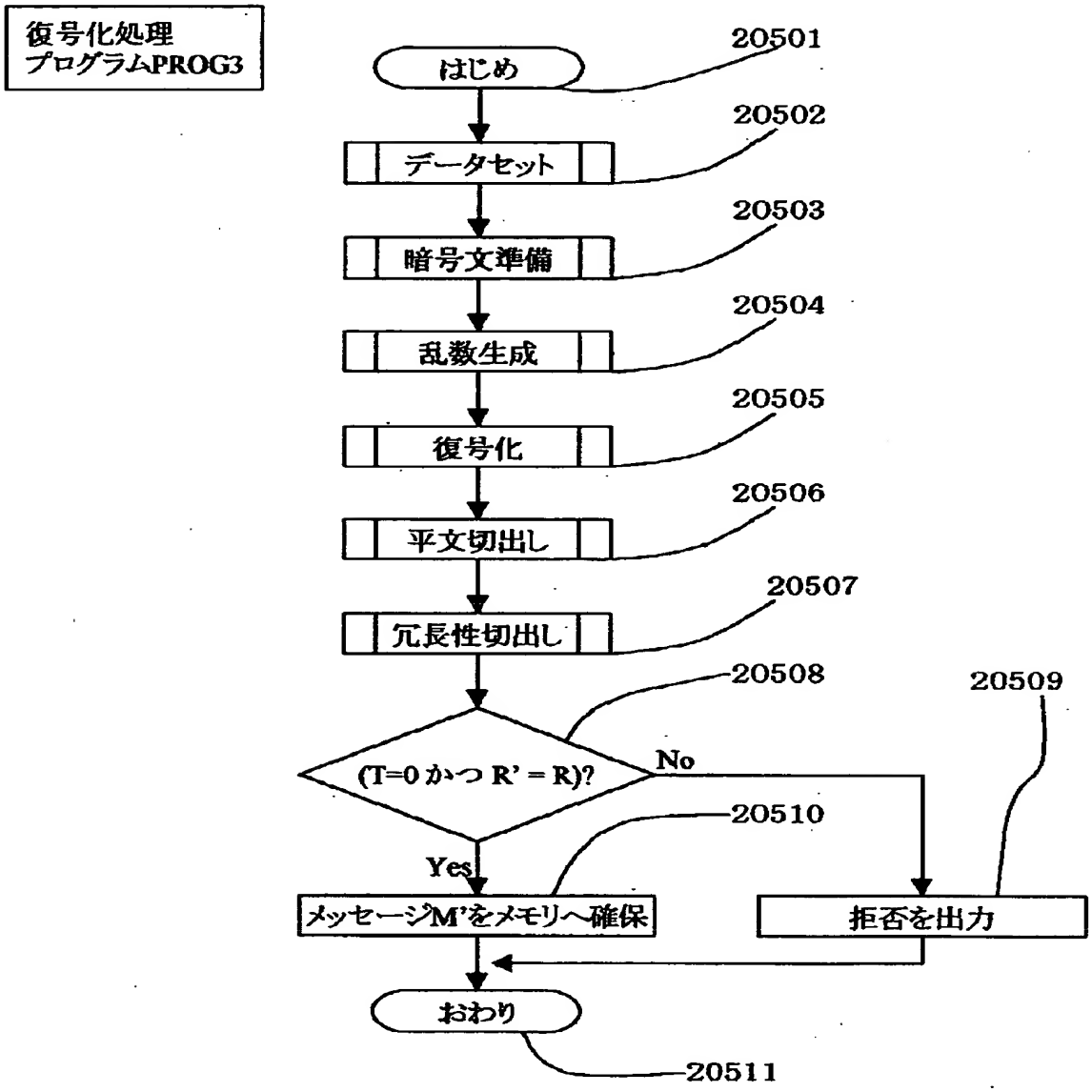
【図４】

図4


暗号化
はじめ
20401
初期値セット: F_0=V
20402
Set i=1
20403
F_i=P_i ^ A_i
20404
C_i= (F_i * B_i) ^ F_{i-1}
20405
i = n?
20406
No
Yes
おわり
20408
i=i+1
20407

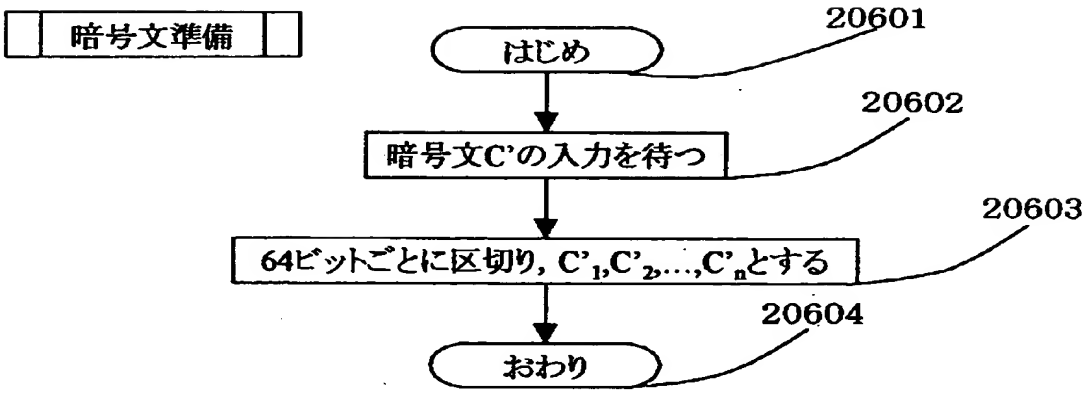
【図５】

図5


復号化処理
プログラムPROG3
はじめ
20501
データセット
20502
暗号文準備
20503
乱数生成
20504
復号化
20505
平文切出し
20506
冗長性切出し
20507
(T=0 かつ R' = R)?
20508
No
20509
Yes
20510
メッセージM'をメモリへ確保
拒否を出力
おわり
20511

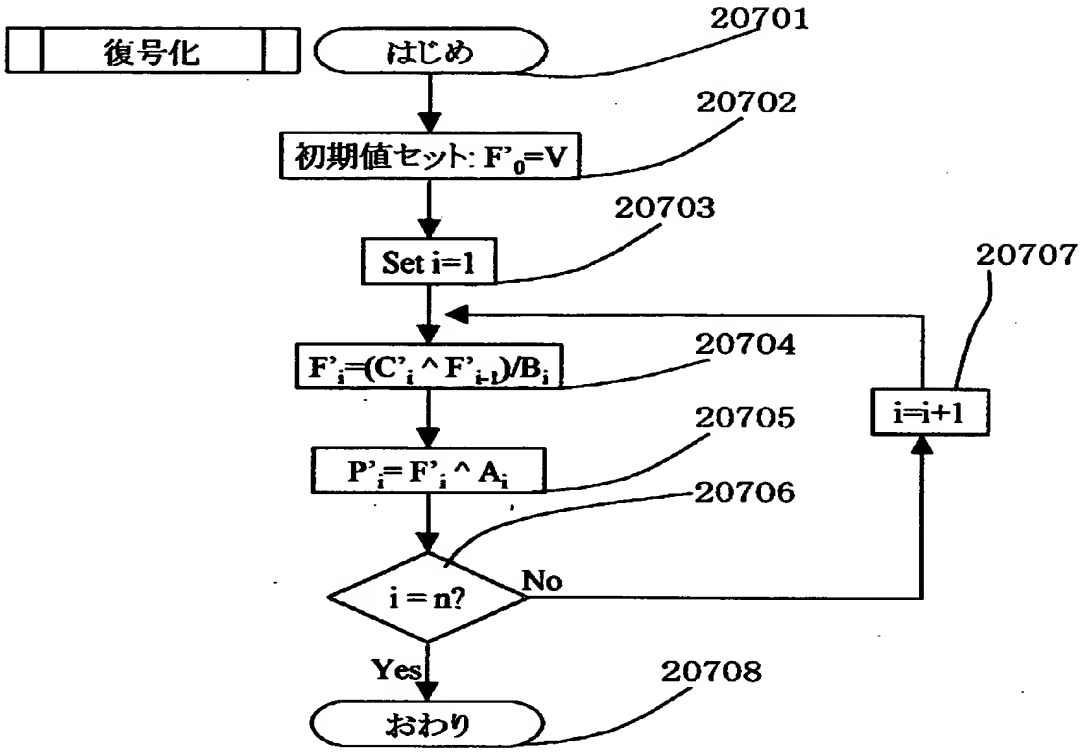
【図６】

図6


暗号文準備
はじめ
20601
暗号文C'の入力を待つ
20602
64ビットごとに区切り, C'1,C'2,...,C'nとする
20603
おわり
20604

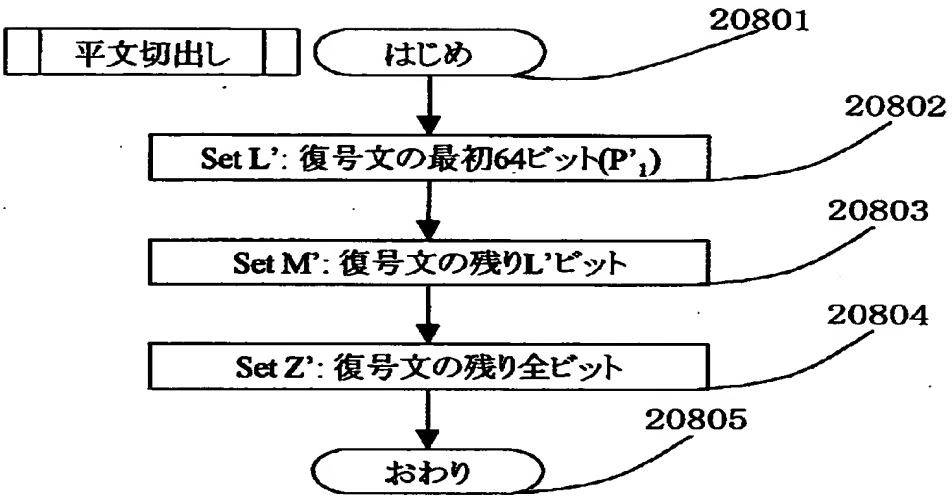
【図７】

図7


復号化
はじめ
20701
初期値セット: F'0=V
20702
Set i=1
20703
F'i=(C'i ^ F'i-1)/Bi
20704
P'i= F'i ^ Ai
20705
i = n?
20706
No
Yes
i=i+1
20707
おわり
20708

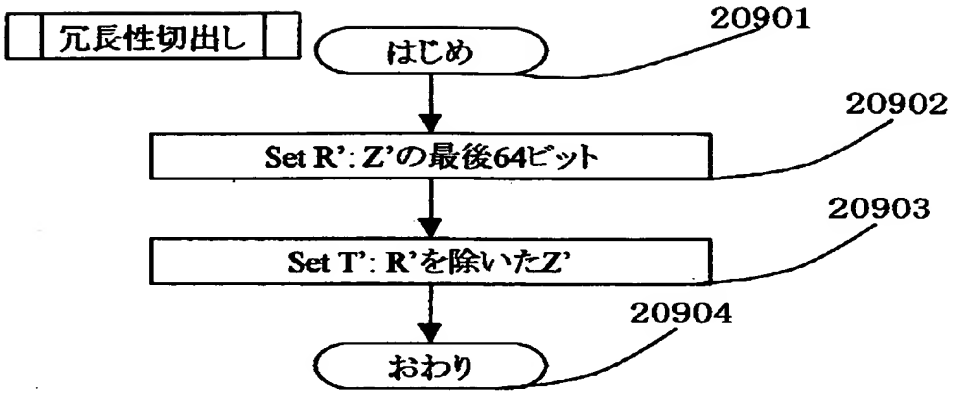
【図８】

図8


平文切出し
はじめ
20801
Set L': 復号文の最初64ビット(P'1)
20802
Set M': 復号文の残りL'ビット
20803
Set Z': 復号文の残り全ビット
20804
おわり
20805

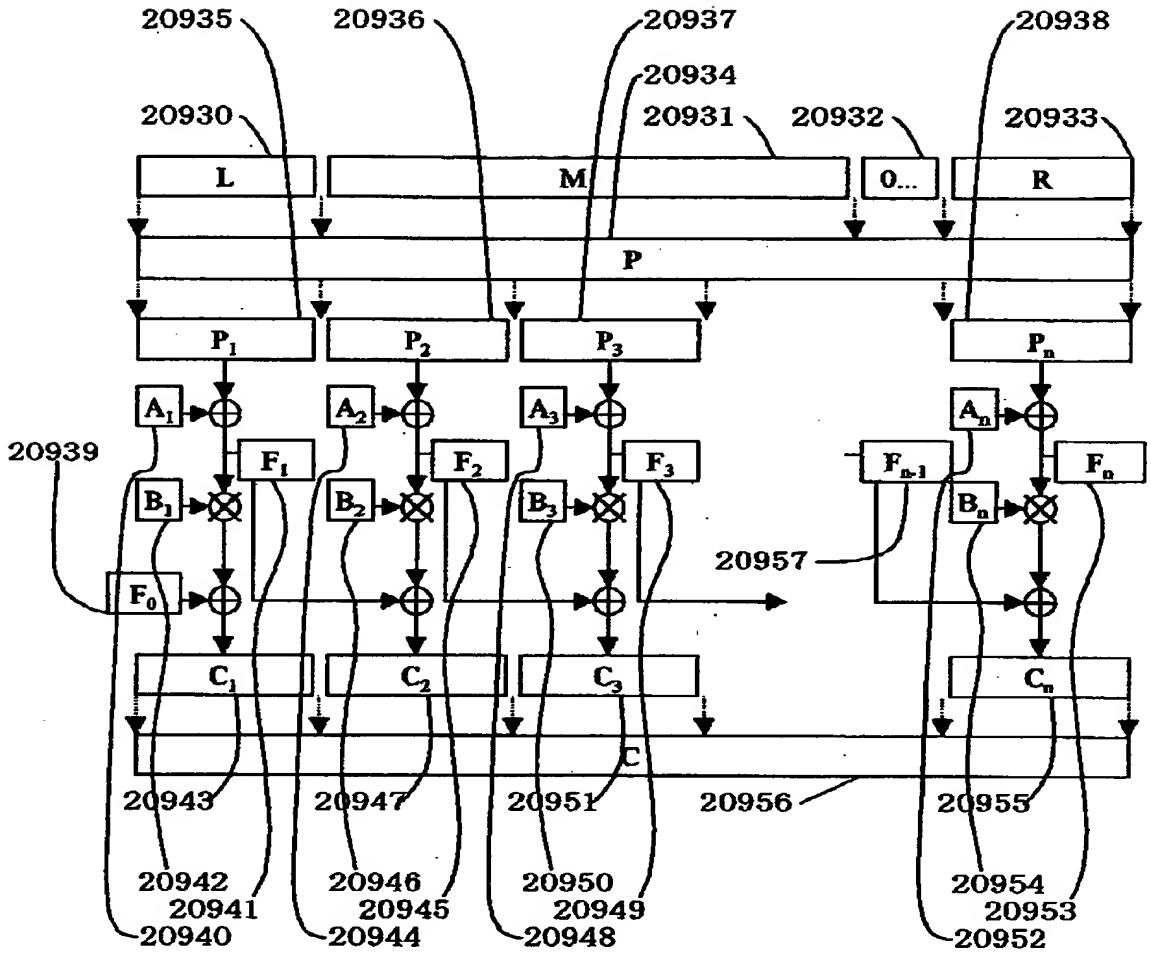
【図９】

図9


冗長性切出し
はじめ
20901
Set R': Z'の最後64ビット
20902
Set T': R'を除いたZ'
20903
おわり
20904

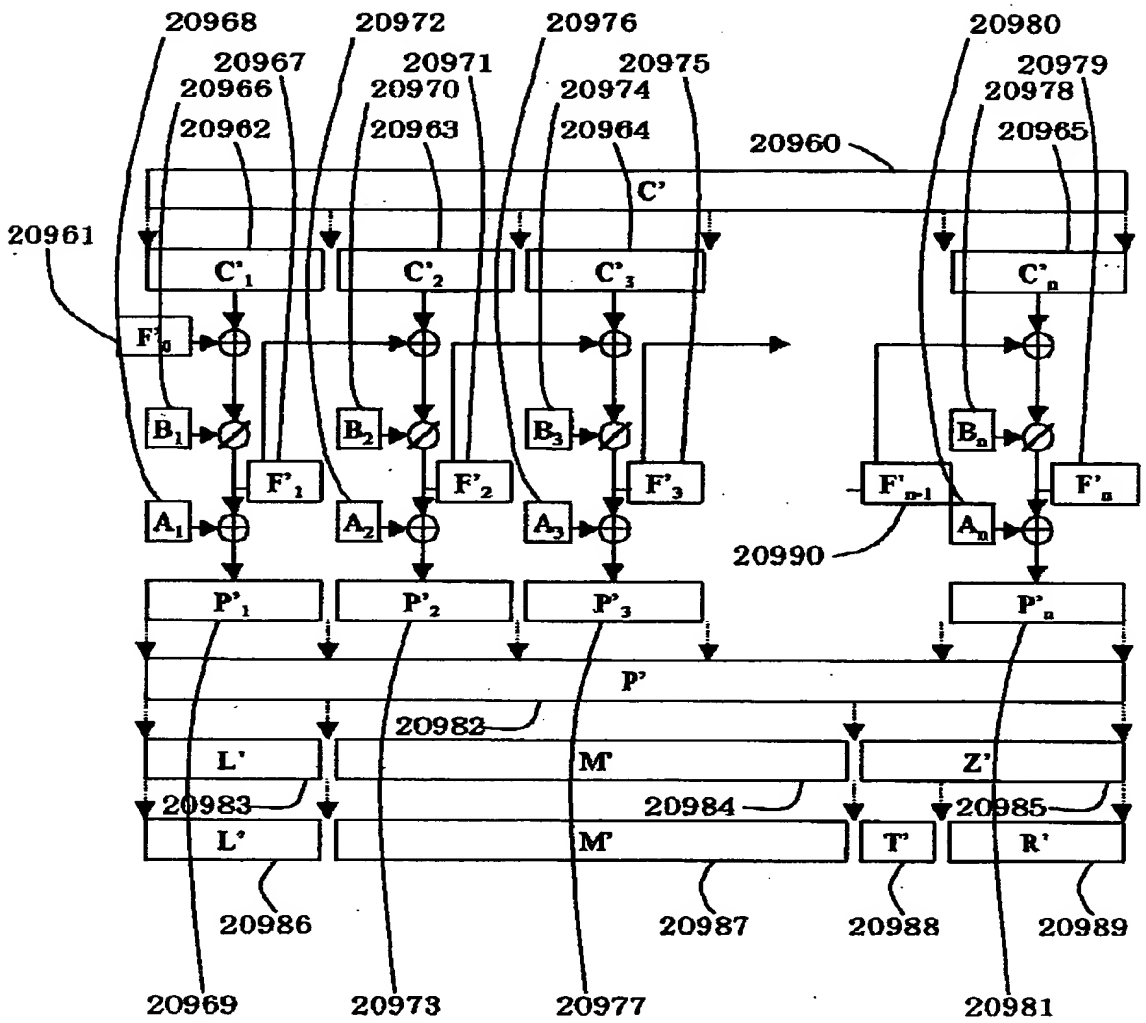
【図１０】

図10


20935
20936
20937
20938
20934
20930
20931
20932
20933
L
M
0...
R
P
P1
P2
P3
Pn
A1
A2
A3
An
20939
F1
F2
F3
Fn-1
Fn
B1
B2
B3
Bn
20957
F0
C1
C2
C3
Cn
C
20943
20947
20951
20956
20955
20942
20941
20940
20946
20945
20944
20950
20949
20948
20954
20953
20952

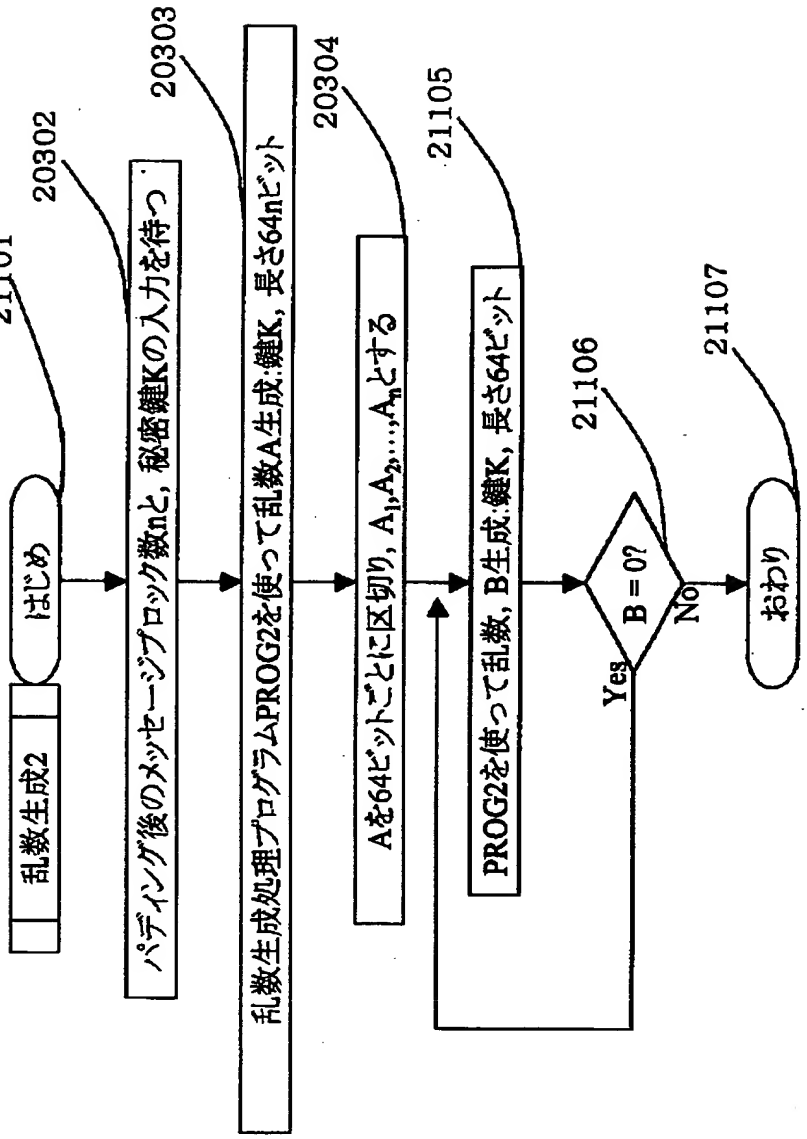
【図１１】

図11


20968
20972
20976
20980
20967
20971
20975
20979
20966
20970
20974
20978
20962
20963
20964
20965
20960
C'
20961
C'1
C'2
C'3
C'n
F'0
B1
B2
B3
Bn
F'1
F'2
F'3
F'n-1
F'n
A1
A2
A3
An
20990
P'1
P'2
P'3
P'n
P'
20982
L'
M'
Z'
20983
20984
20985
L'
M'
T'
R'
20986
20987
20988
20989
20969
20973
20977
20981

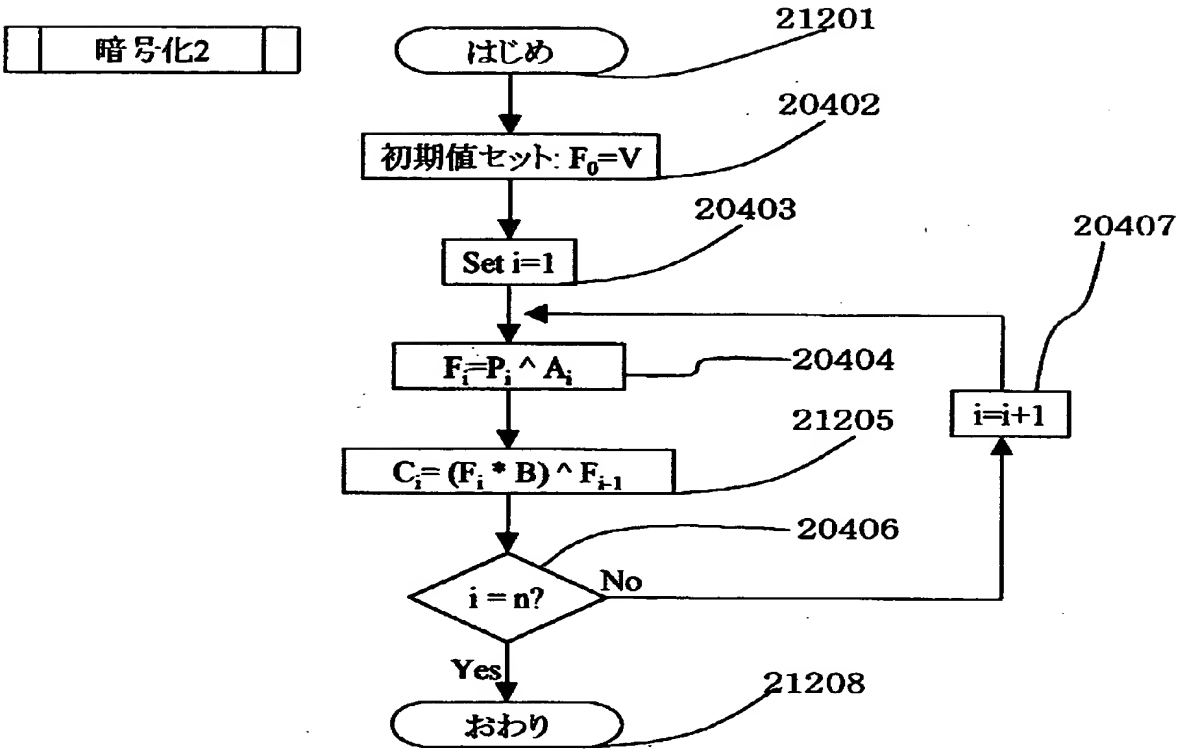
【図１２】

図12


乱数生成2
はじめ
21101
パディング後のメッセージブロック数nと，秘密鍵Kの入力を待つ
20302
乱数生成処理プログラムPROG2を使って乱数A生成:鍵K，長さ64nビット
20303
Aを64ビットごとに区切り，A1,A2,...,Anとする
20304
PROG2を使って乱数，B生成:鍵K，長さ64ビット
21105
B = 0?
21106
Yes
No
おわり
21107

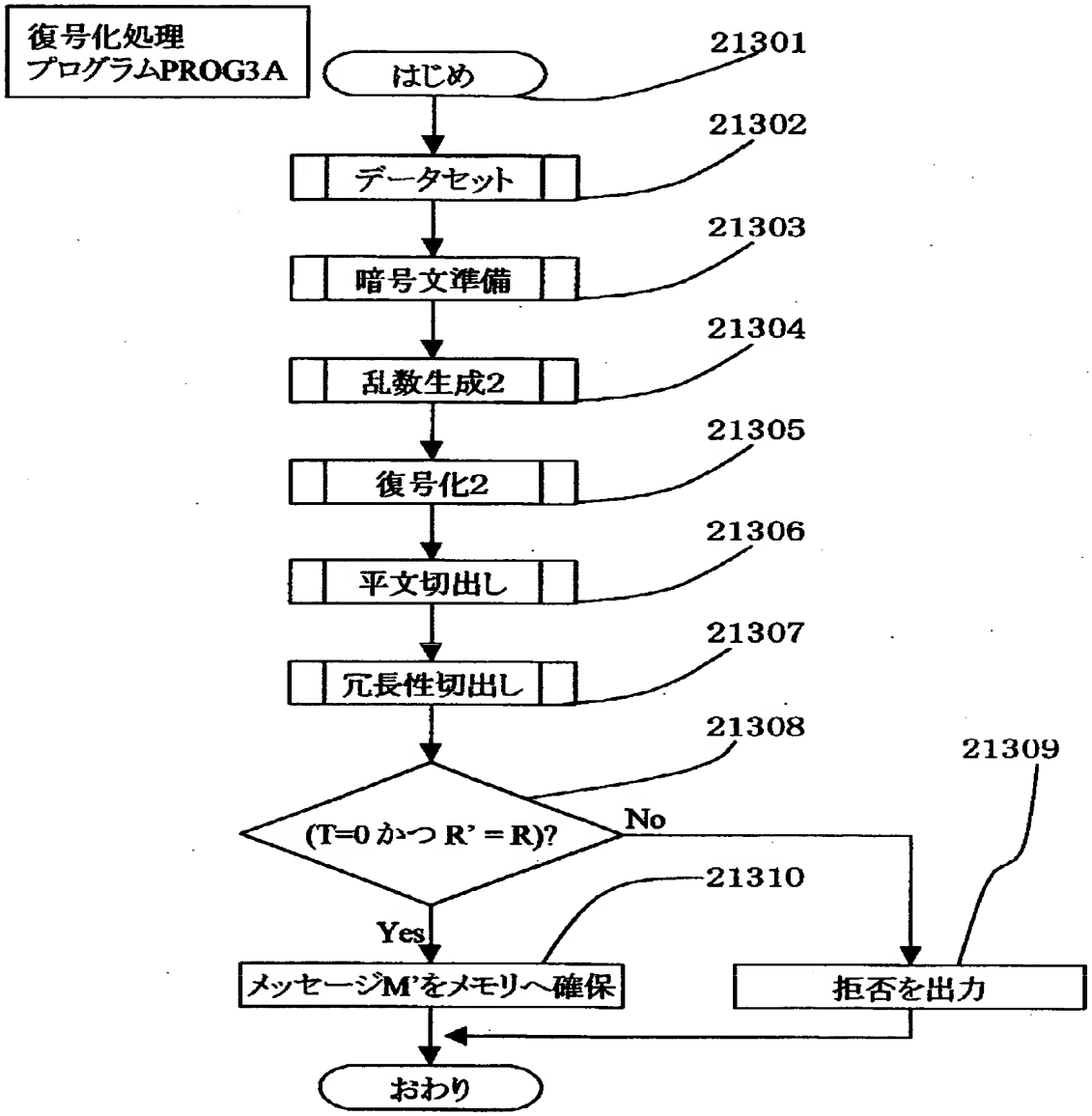
【図１３】

図13


暗号化2
はじめ
21201
初期値セット: F_0=V
20402
Set i=1
20403
20407
F_i=P_i ^ A_i
20404
i=i+1
21205
C_i= (F_i * B) ^ F_{i-1}
20406
i = n?
No
Yes
おわり
21208

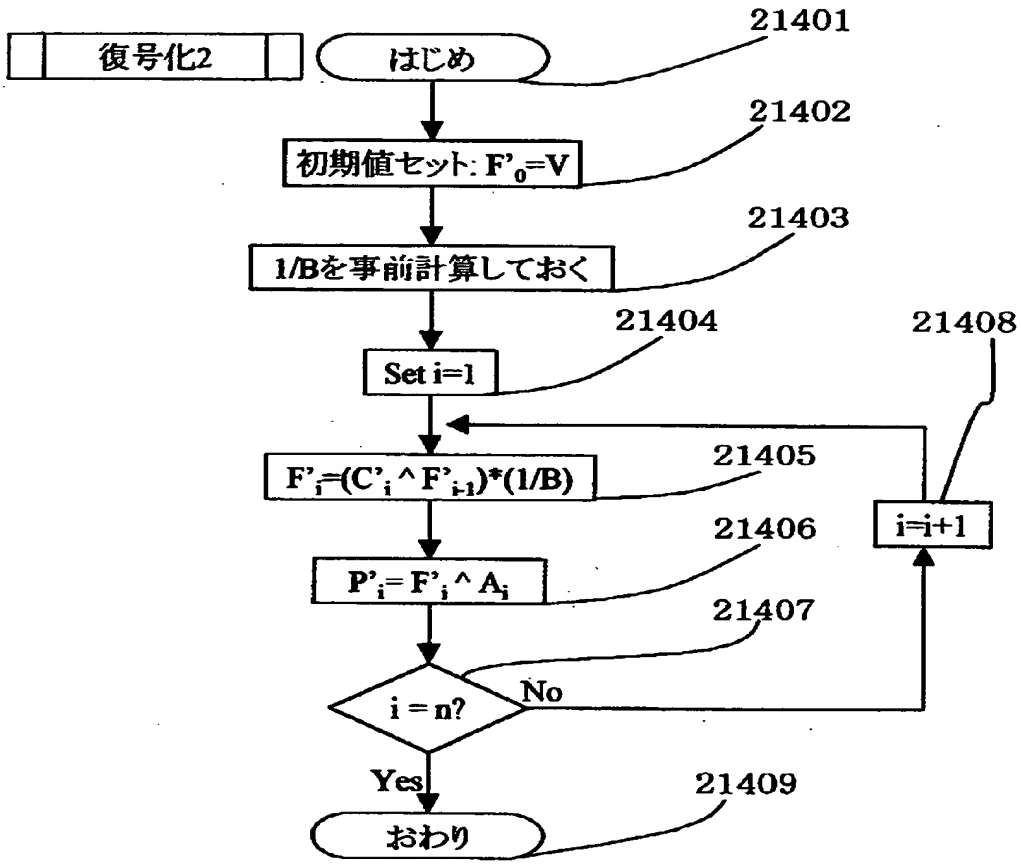
【図１４】

図14


復号化処理
プログラムPROG3A
はじめ
21301
データセット
21302
暗号文準備
21303
乱数生成2
21304
復号化2
21305
平文切出し
21306
冗長性切出し
21307
(T=0 かつ R' = R)?
21308
No
21309
Yes
21310
メッセージM'をメモリへ確保
拒否を出力
おわり

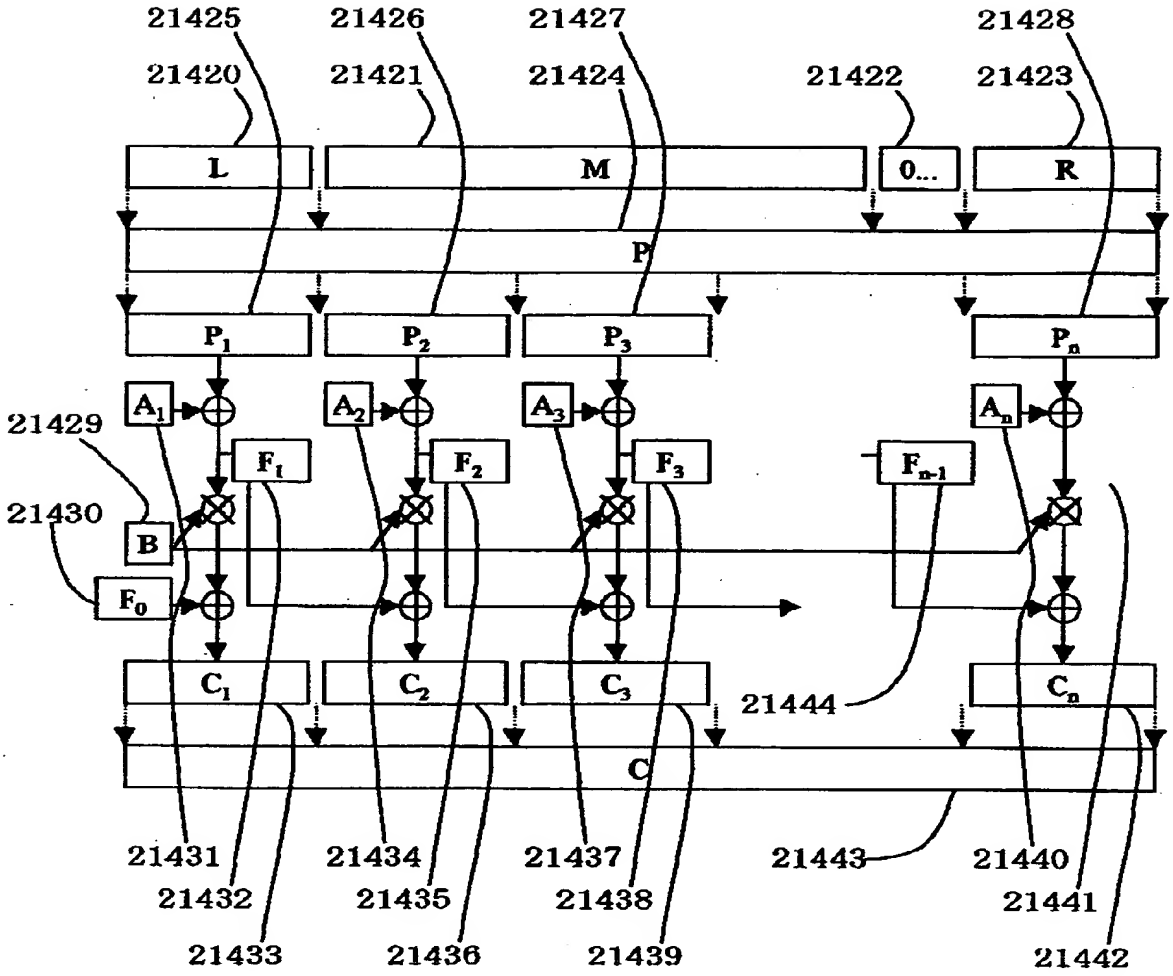
【図１５】

図15


復号化2
はじめ
21401
初期値セット: F'0=V
21402
1/Bを事前計算しておく
21403
Set i=1
21404
F'i=(C'i ^ F'i-1)*(1/B)
21405
P'i= F'i ^ Ai
21406
i = n?
21407
No
Yes
おわり
21409
i=i+1
21408

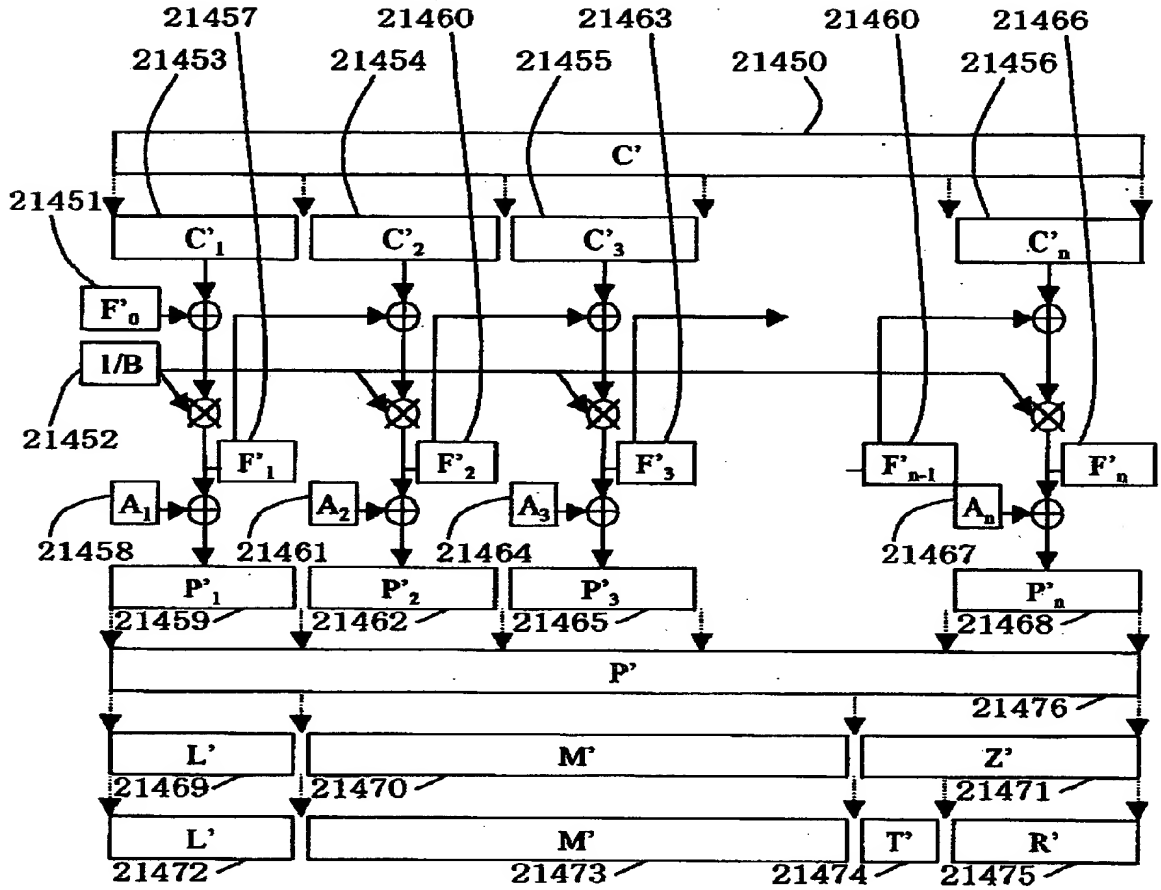
【図１６】

図16


21425
21426
21427
21428
21420
21421
21424
21422
21423
L
M
0...
R
P
P1
P2
P3
Pn
A1
A2
A3
An
21429
F1
F2
F3
Fn-1
21430
B
F0
C1
C2
C3
Cn
21444
C
21431
21434
21437
21443
21440
21432
21435
21438
21441
21433
21436
21439
21442

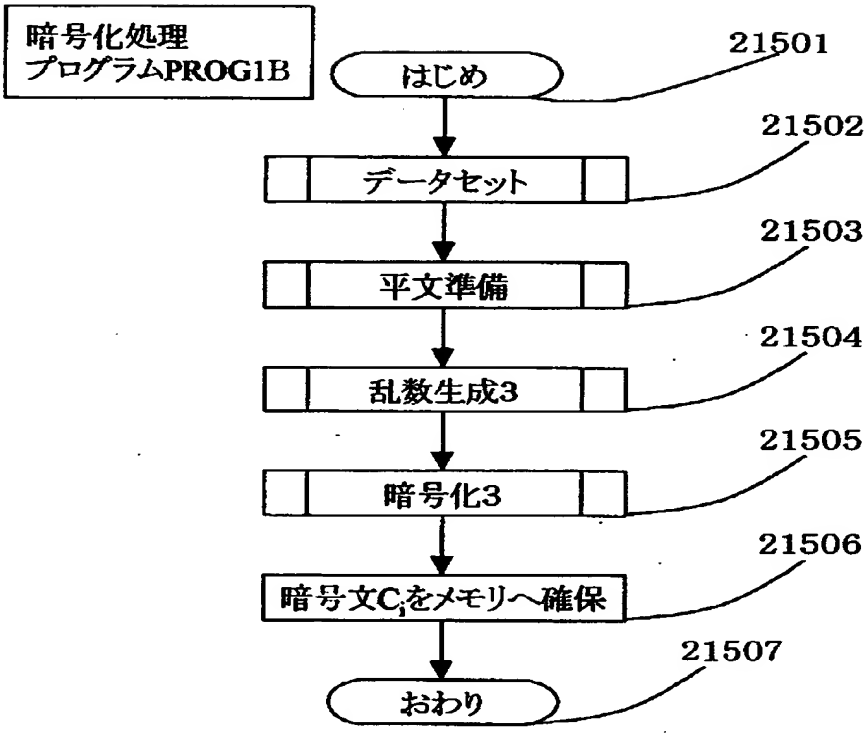
【図１７】

図17


21457
21460
21463
21460
21466
21453
21454
21455
21450
21456
C'
21451
C'1
C'2
C'3
C'n
F'0
1/B
21452
F'1
F'2
F'3
F'n-1
F'n
A1
A2
A3
An
21458
21461
21464
21467
P'1
P'2
P'3
P'n
21459
21462
21465
21468
P'
21476
L'
M'
Z'
21469
21470
21471
L'
M'
T'
R'
21472
21473
21474
21475

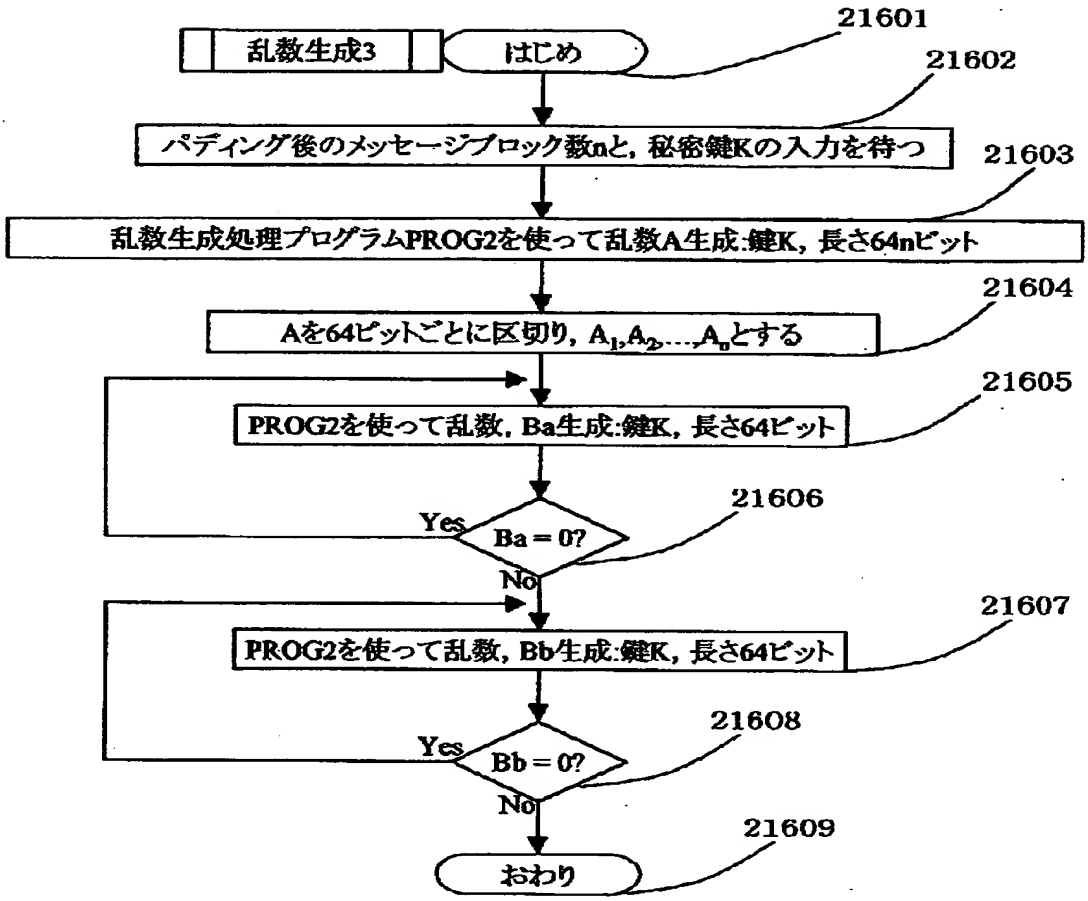
【図１８】

図18


暗号化処理
プログラムPROG1B
はじめ
21501
データセット
21502
平文準備
21503
乱数生成3
21504
暗号化3
21505
暗号文C_iをメモリへ確保
21506
おわり
21507

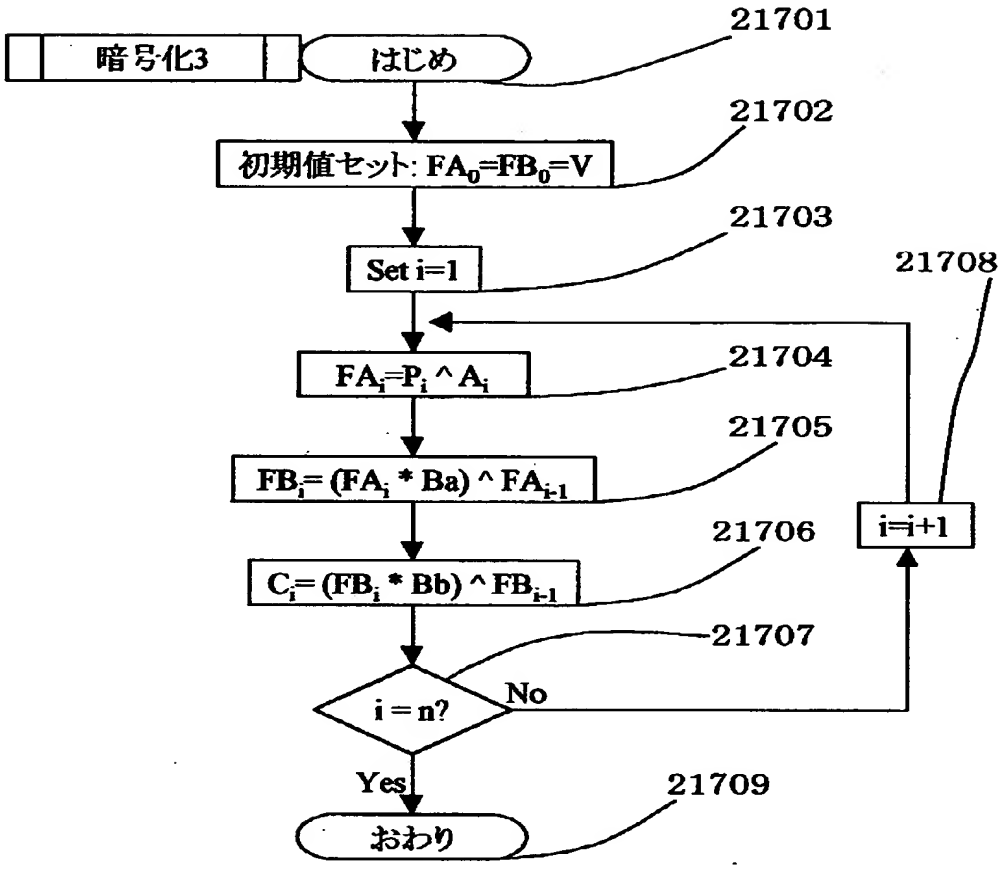
【図１９】

図19


乱数生成3
はじめ
21601
21602
パディング後のメッセージブロック数nと，秘密鍵Kの入力を待つ
21603
乱数生成処理プログラムPROG2を使って乱数A生成:鍵K，長さ64nビット
21604
Aを64ビットごとに区切り，A1,A2,...,Anとする
21605
PROG2を使って乱数，Ba生成:鍵K，長さ64ビット
21606
Yes
Ba = 0?
No
21607
PROG2を使って乱数，Bb生成:鍵K，長さ64ビット
21608
Yes
Bb = 0?
No
21609
おわり

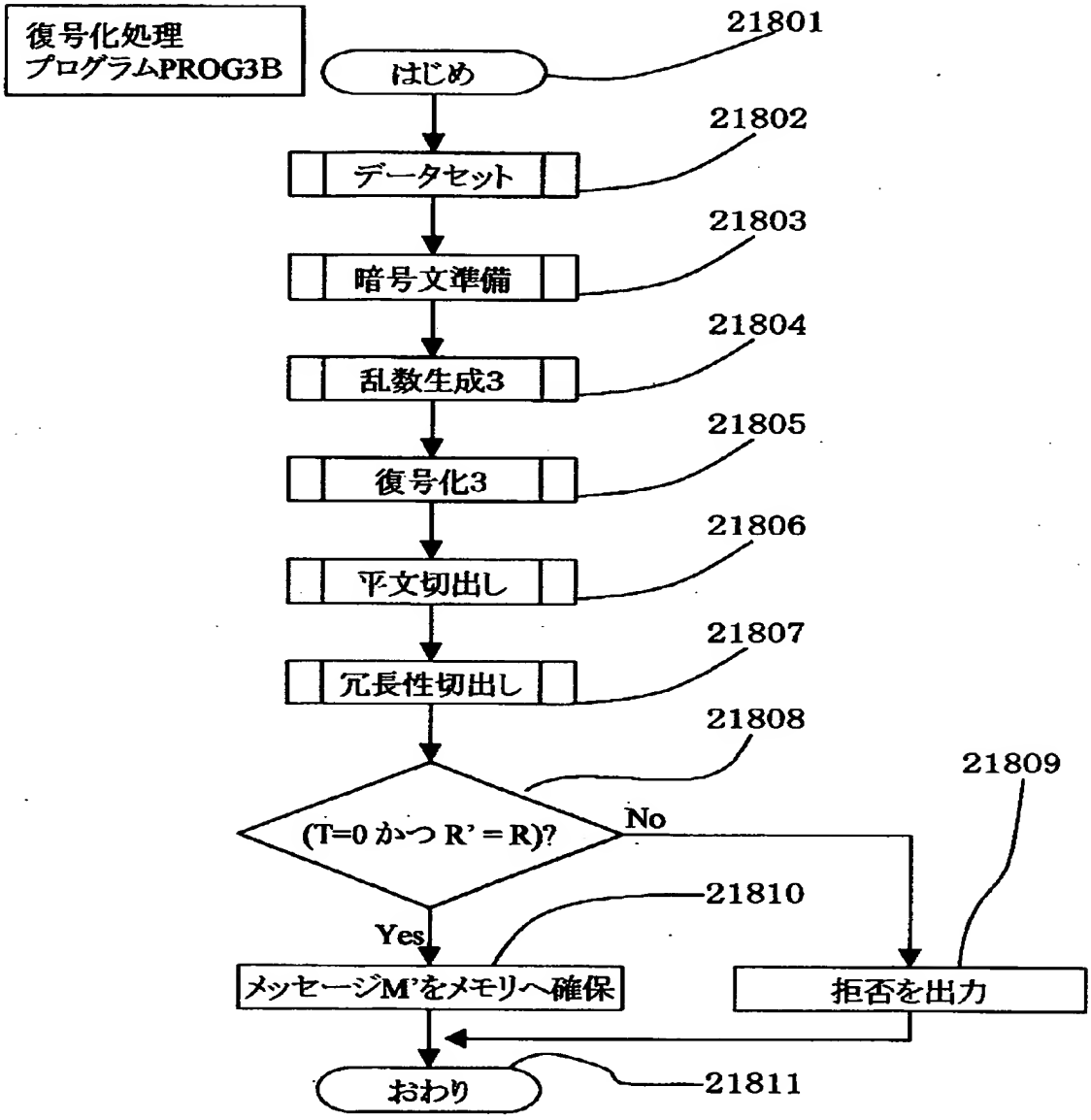
【図２０】

図20


暗号化3
はじめ
21701
初期値セット: $FA_0=FB_0=V$
21702
Set i=1
21703
$FA_i=P_i \wedge A_i$
21704
$FB_i=(FA_i * Ba) \wedge FA_{i-1}$
21705
$C_i=(FB_i * Bb) \wedge FB_{i-1}$
21706
i = n?
21707
No
Yes
おわり
21709
i=i+1
21708

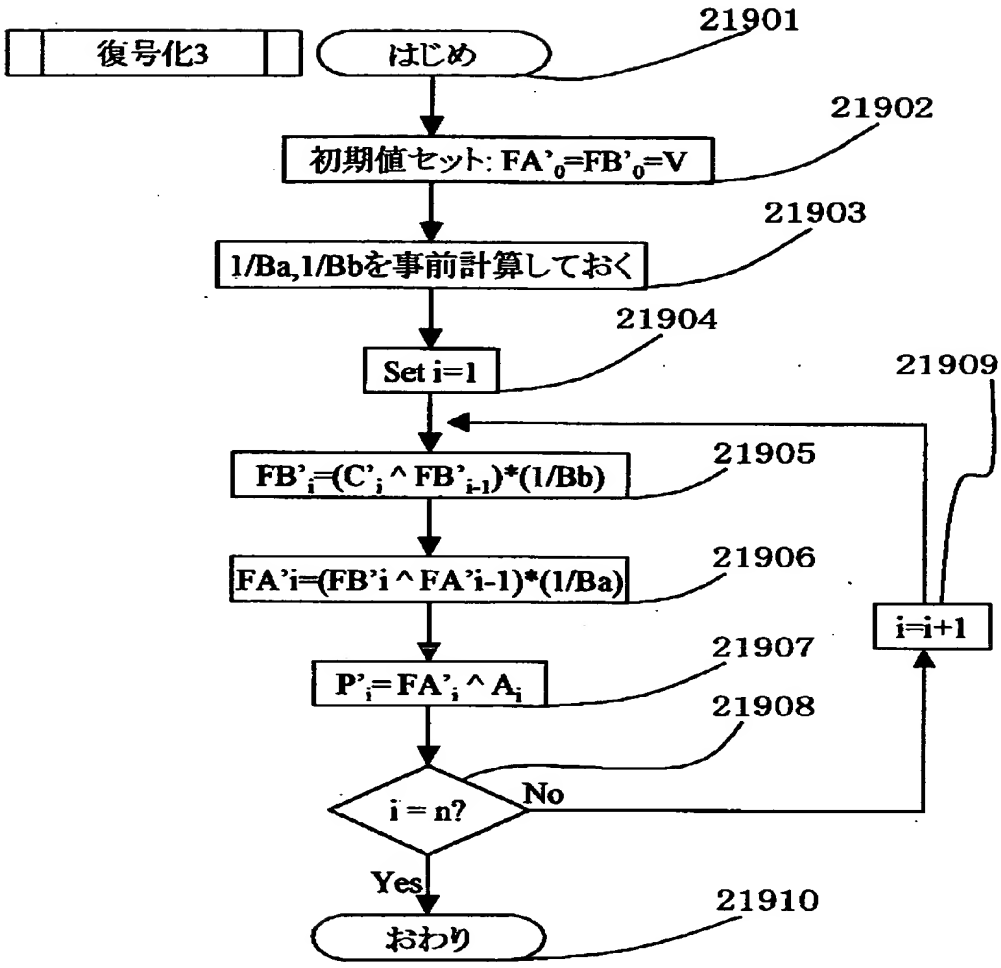
【図２１】

図21


復号化処理
プログラムPROG3B
はじめ
21801
データセット
21802
暗号文準備
21803
乱数生成3
21804
復号化3
21805
平文切出し
21806
冗長性切出し
21807
(T=0 かつ R' = R)?
21808
No
21809
Yes
21810
メッセージM'をメモリへ確保
拒否を出力
おわり
21811

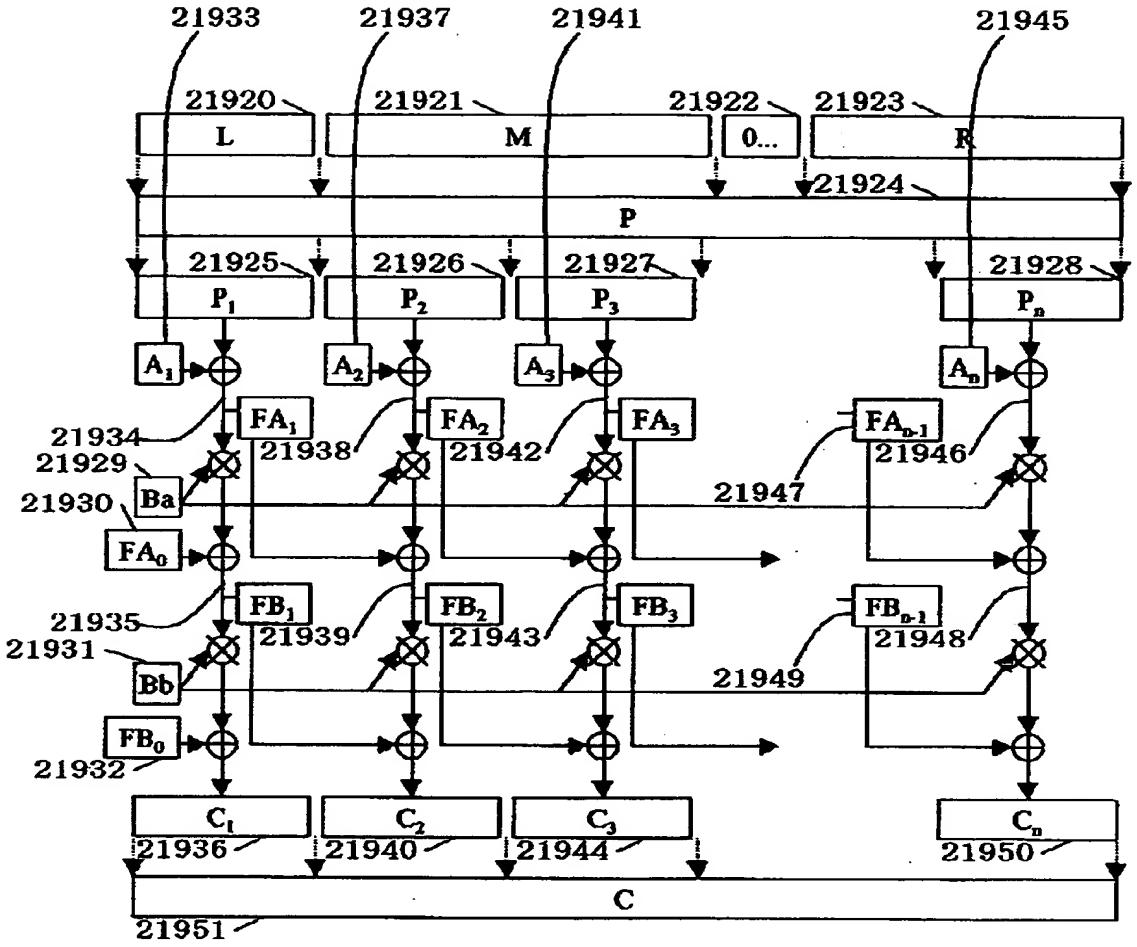
【図２２】

図22


復号化3
はじめ
21901
初期値セット: FA'0=FB'0=V
21902
1/Ba,1/Bbを事前計算しておく
21903
Set i=1
21904
FB'i=(C'i ^ FB'i-1)*(1/Bb)
21905
FA'i=(FB'i ^ FA'i-1)*(1/Ba)
21906
P'i= FA'i ^ Ai
21907
i = n?
21908
No
Yes
i=i+1
21909
おわり
21910

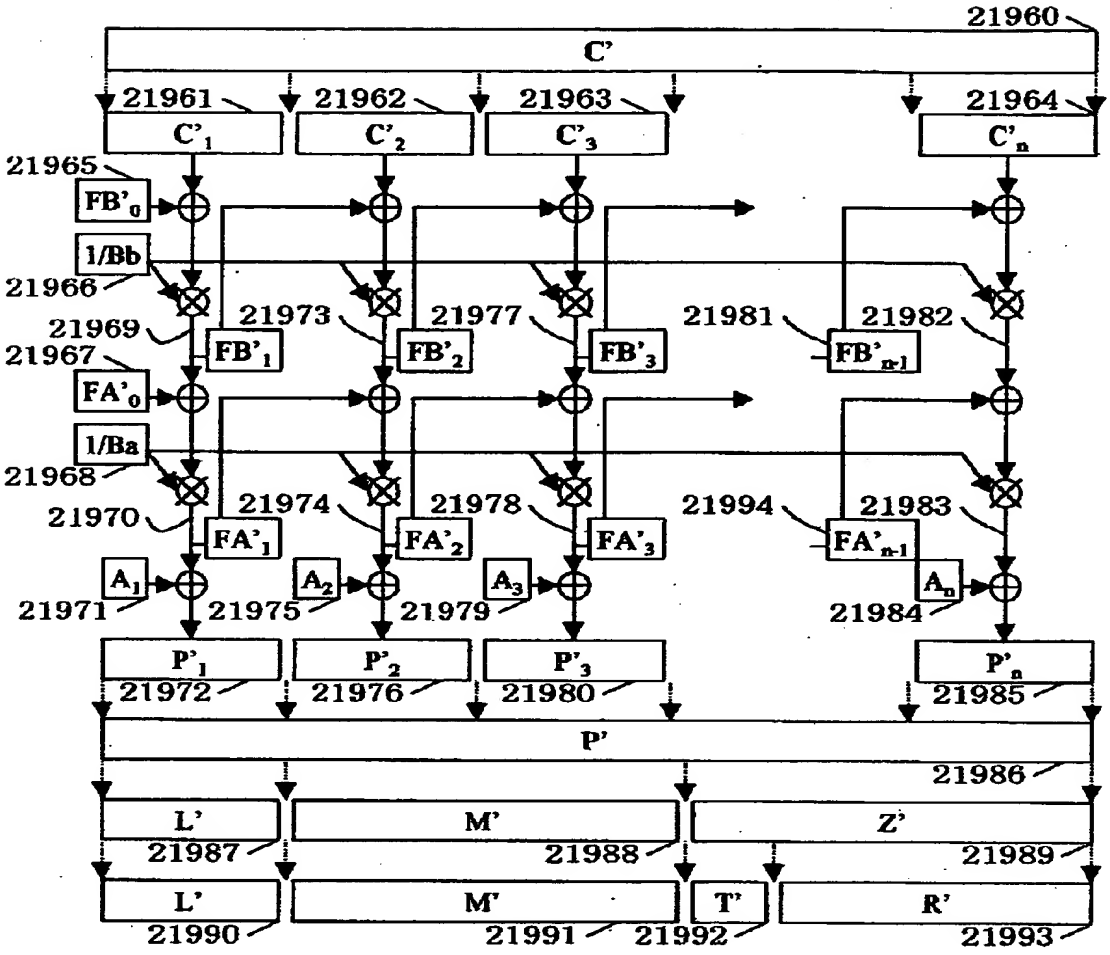
【図２３】

図23


21933
21937
21941
21945
21920
21921
21922
21923
L
M
0...
R
21924
P
21925
21926
21927
21928
P1
P2
P3
Pn
A1
A2
A3
An
FA1
FA2
FA3
FAn-1
21934
21938
21942
21946
21929
21947
21930
Ba
FA0
FB1
FB2
FB3
FBn-1
21935
21939
21943
21948
21931
21949
Bb
FB0
21932
C1
C2
C3
Cn
21936
21940
21944
21950
C
21951

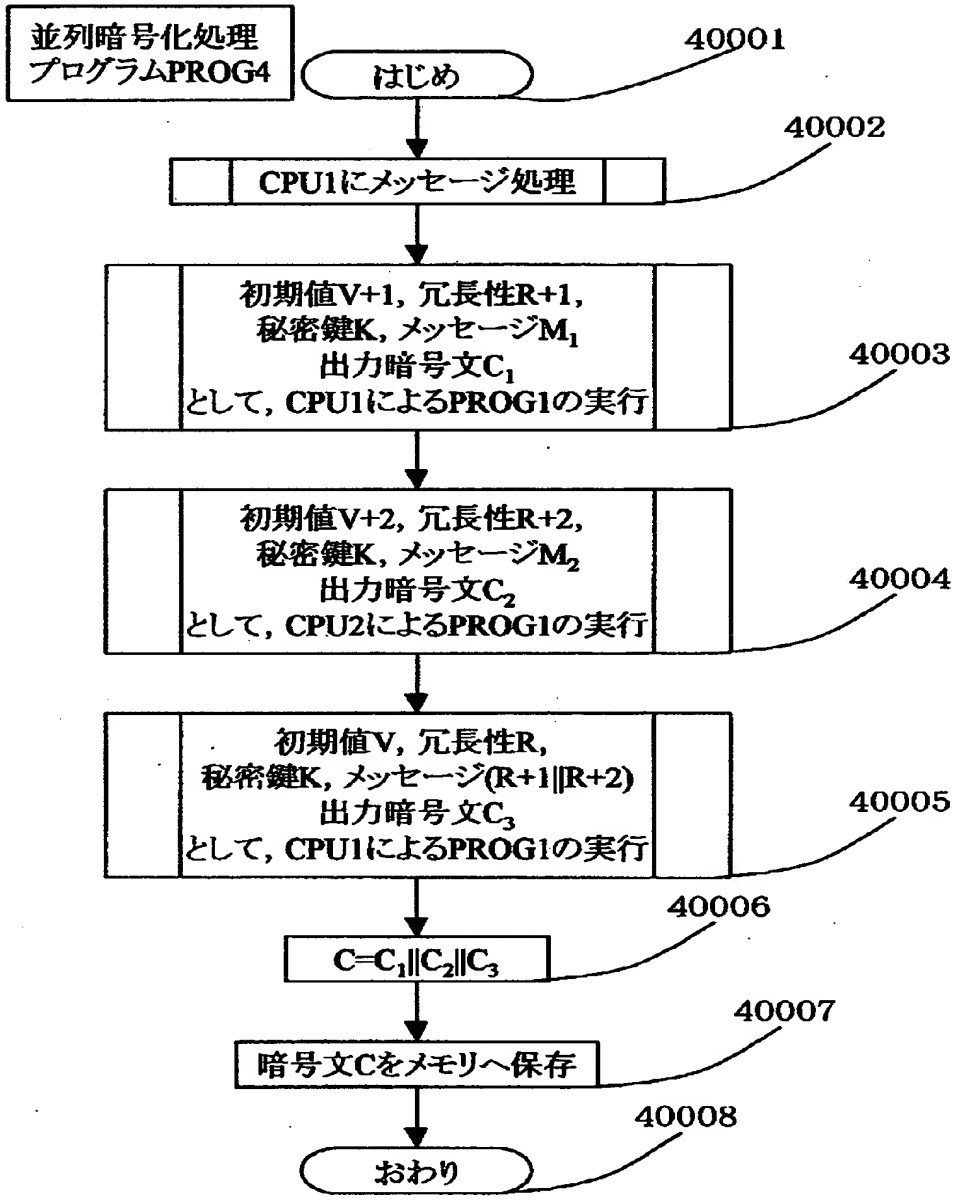
【図２４】

図24


21960
C'
21961
21962
21963
21964
C'1
C'2
C'3
C'n
21965
FB'0
1/Bb
21966
21969
21973
21977
21981
21982
21967
FB'1
FB'2
FB'3
FB'n-1
FA'0
1/Ba
21968
21974
21978
21994
21983
21970
FA'1
FA'2
FA'3
FA'n-1
A1
A2
A3
An
21971
21975
21979
21984
P'1
P'2
P'3
P'n
21972
21976
21980
21985
P'
21986
L'
M'
Z'
21987
21988
21989
L'
M'
T'
R'
21990
21991
21992
21993

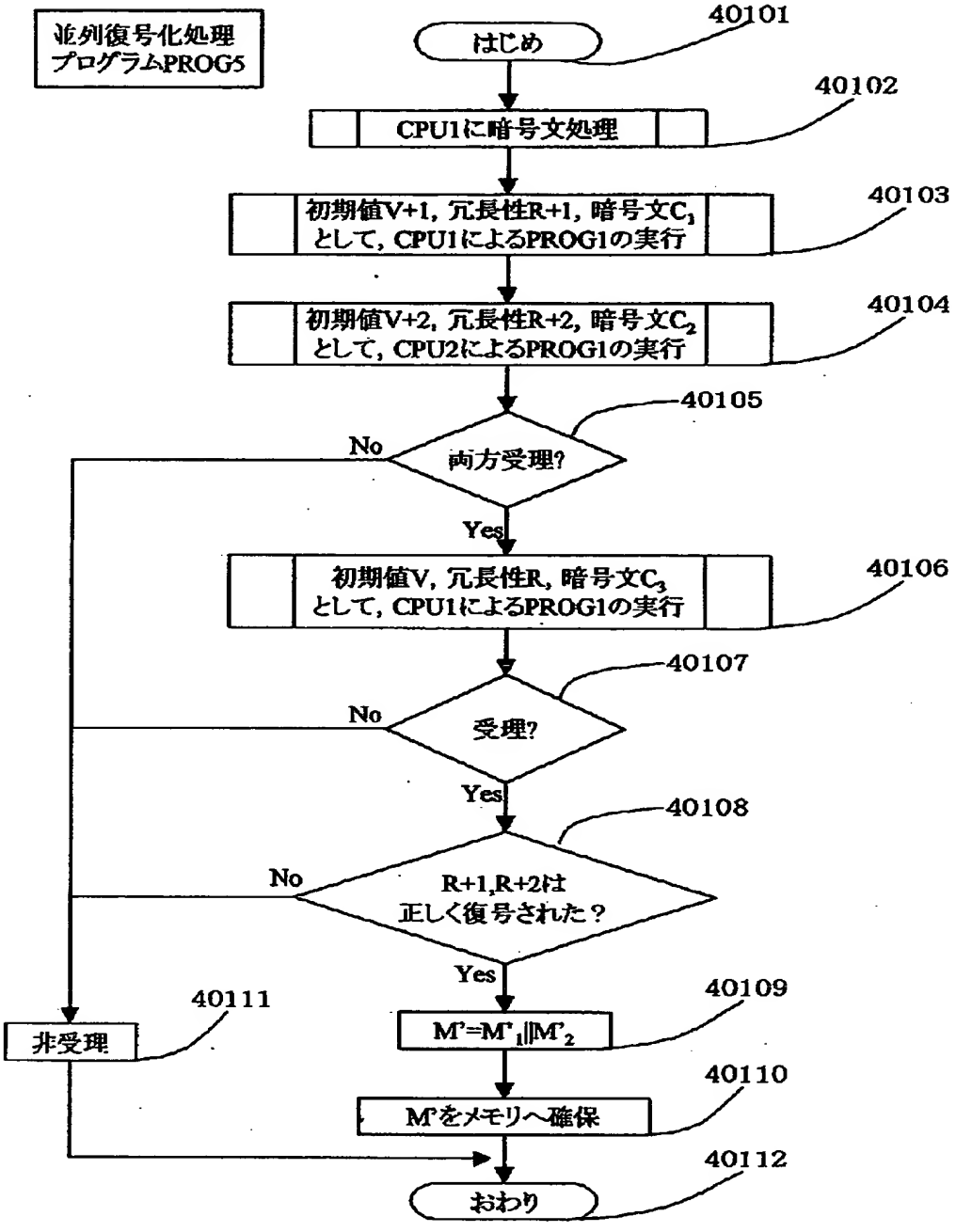
【図２５】

図25


並列暗号化処理
プログラムPROG4
はじめ
40001
CPU1にメッセージ処理
40002
初期値V+1, 冗長性R+1,
秘密鍵K, メッセージM_1
出力暗号文C_1
として, CPU1によるPROG1の実行
40003
初期値V+2, 冗長性R+2,
秘密鍵K, メッセージM_2
出力暗号文C_2
として, CPU2によるPROG1の実行
40004
初期値V, 冗長性R,
秘密鍵K, メッセージ(R+1||R+2)
出力暗号文C_3
として, CPU1によるPROG1の実行
40005
$C=C_1||C_2||C_3$
40006
暗号文Cをメモリへ保存
40007
おわり
40008

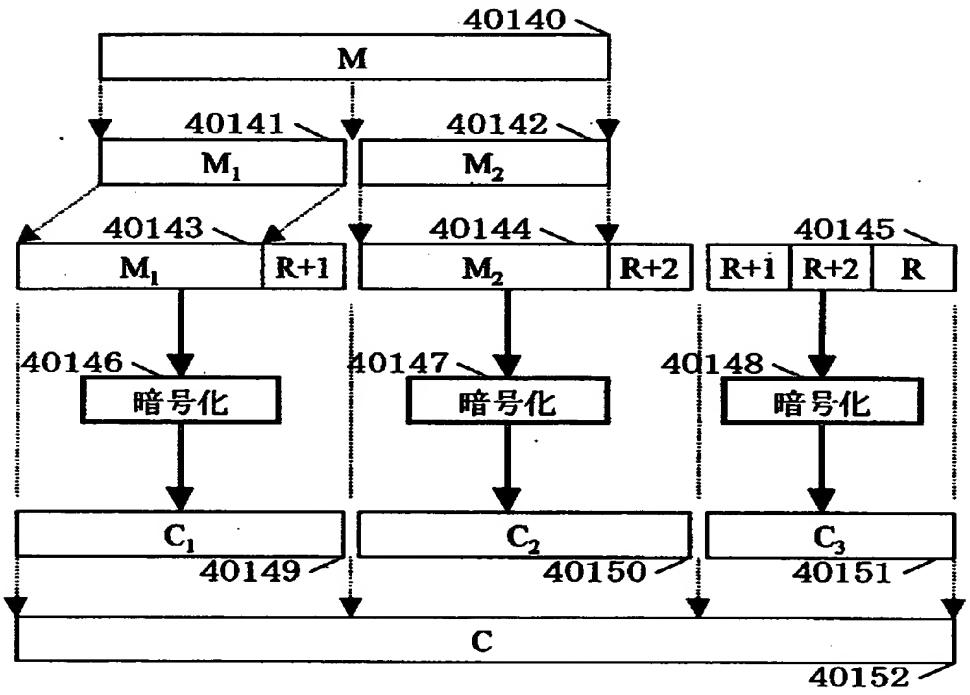
【図２６】

図26


並列復号化処理
プログラムPROG5
はじめ
40101
CPU1に暗号文処理
40102
初期値V+1, 冗長性R+1, 暗号文C1
として, CPU1によるPROG1の実行
40103
初期値V+2, 冗長性R+2, 暗号文C2
として, CPU2によるPROG1の実行
40104
両方受理?
40105
No
Yes
初期値V, 冗長性R, 暗号文C3
として, CPU1によるPROG1の実行
40106
受理?
40107
No
Yes
R+1,R+2は
正しく復号された?
40108
No
Yes
M'=M'1||M'2
40109
M'をメモリへ確保
40110
非受理
40111
おわり
40112

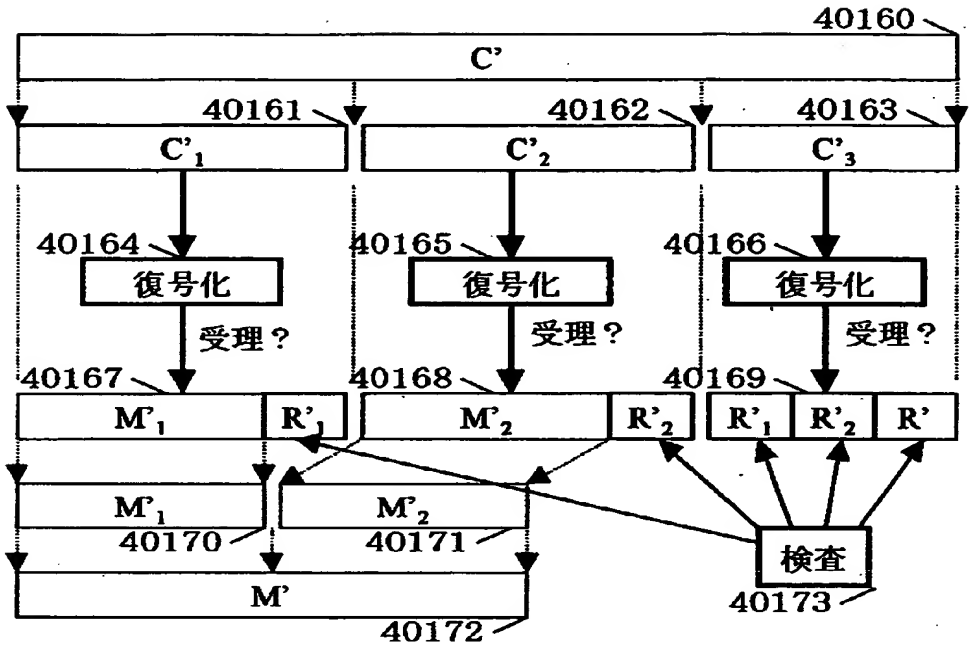
【図２７】

図27


40140
M
40141
40142
M1
M2
40143
40144
40145
M1
R+1
M2
R+2
R+1
R+2
R
40146
40147
40148
暗号化
暗号化
暗号化
C1
C2
C3
40149
40150
40151
C
40152

【図２８】

図28


40160
C'
40161
40162
40163
C'1
C'2
C'3
40164
40165
40166
復号化
復号化
復号化
受理？
受理？
受理？
40167
40168
40169
M'1
R'1
M'2
R'2
R'1
R'2
R'
M'1
M'2
40170
40171
検査
M'
40173
40172

【書類名】　要約書

【要約】

【課題】

高速で,並列度の高い暗号処理と改ざん検出を実現できる共通鍵暗号技術を提案すること。

【解決手段】

暗号化したいデータに冗長性を付加し,その長さよりも真に長い長さの鍵ストリームにより暗号化し,暗号文を生成する。復号化では冗長性の復元を確認することで通信中の改ざんを検出することができる。改ざんの成功確率が評価されており,ブロック暗号よりも高速で,並列度が高く,実装規模を小さくできる。

【選択図】　図１

出　願　人　履　歴　情　報

識別番号　　〔０００００５１０８〕

１．変更年月日　　１９９０年　８月３１日
［変更理由］　新規登録
住　所　東京都千代田区神田駿河台４丁目６番地
氏　名　株式会社日立製作所